FEBRUARY 1976 $1.50

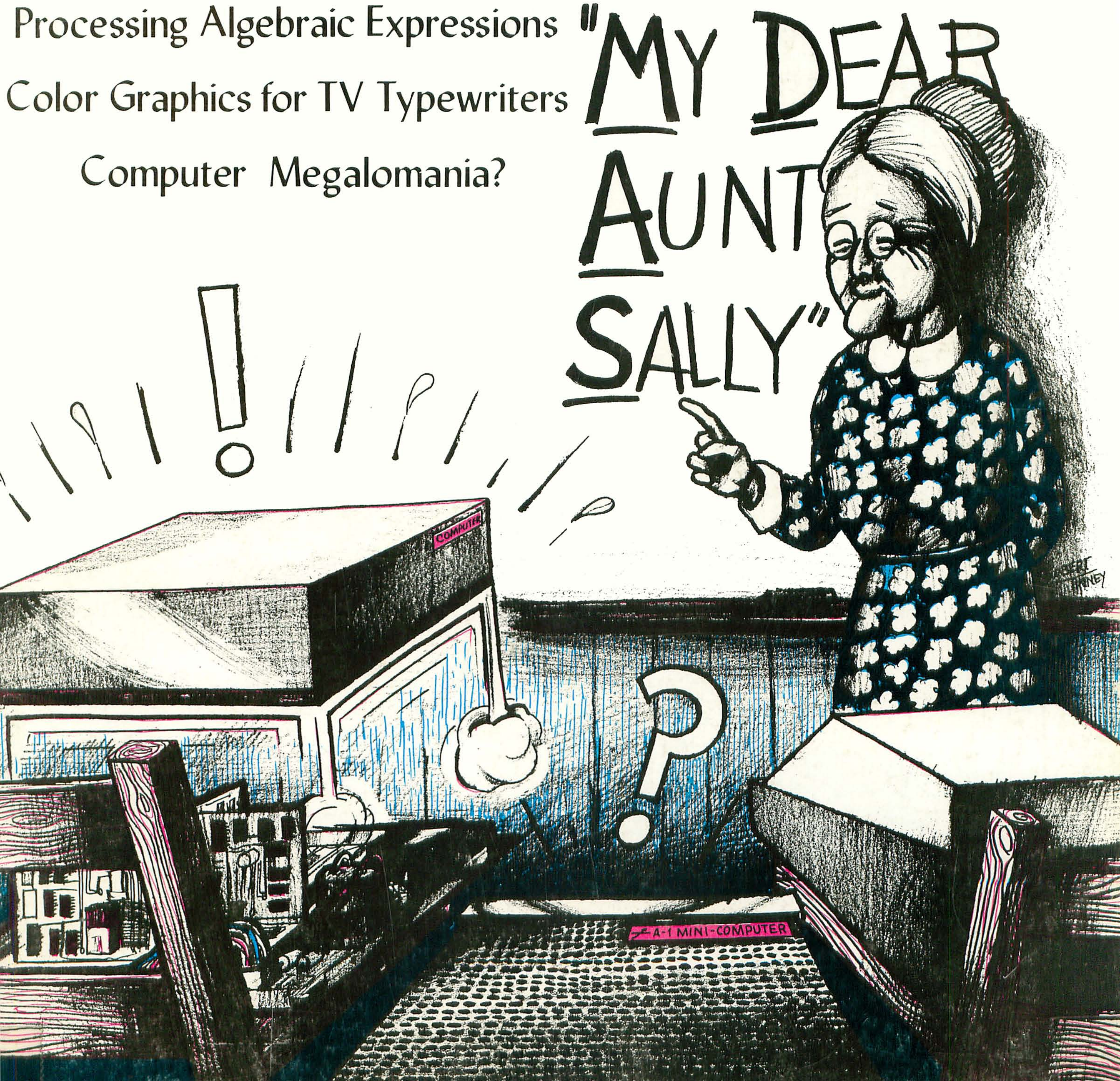
BYTE

the small systems journal

Processing Algebraic Expressions

Color Graphics for TV Typewriters

Computer Megalomania?



THINKING ABOUT A "6800" TYPE COMPUTER?

It seems that a great many people came to the same conclusion that we did here at SwTPC. The M6800 is an outstanding processor and makes a great computer — "BUT" — Not all computers using the M6800 processor are the same. May we suggest that you consider the following features when you make your choice.

IT IS A COMPLETE 6800 SYSTEM?

You cannot get all of the advantages of the 6800 system with only the processor chip. Unless the whole 6800 family of chips is used you cannot possibly get all of the versitility and superior performance that the system is capable of providing. If for instance the design does not use the MC6820 parallel and the MC6850 serial integrated circuits for interfacing, you lose the programmable interface feature that makes it so easy to interconnect the computer system with outside devices such as terminals, printers, disks, etc.

IS THE SOFTWARE COMPATABLE OR UNIQUE?

If the design does not use the "Motorola" Mikbug® ROM, then the software and programs that will run on the system are probably unique to that particular brand of computer. SwTPC uses the standard Motorola MCM6830L7 ROM. This provides automatic loading and an operating system that is compatable with other systems using the standard widely sold Motorola evaluation set. As an owner of our 6800 computer system, you are eligible for membership in the Motorola Users Group. If you join you have access to a library of programs that will run on your system. Editor and assembler programs are available directly from SwTPC.

CAN THE SYSTEM BE EXPANDED AT A REASONABLE COST?

Some of the limited systems being offered at lower prices can be expanded only with difficulty. Check the amount of memory that can be added and at what cost. How many additional interfaces can be added, if any. How much of the above can be run off of the power supply provided with the system? The SwTPC 6800 can be expanded up to 16K words of memory in the standard cabinet and with the power supply provided. It may also be expanded up to eight interface (I/0) boards for external devices by simply plugging in the cards. Memory is $125.00 for each 4,096 words of expansion and interface cards are only $35.00 for serial or parallel types.

Memory expansion will be essential if you ever intend to use a resident assembler, or higher level languages such as APL or BASIC on your system. Assembler programs typically require a minimum of 4,096 words of memory and higher level languages require even more.

HOW DO YOU ENTER AND READ DATA?

Let's hope it is by way of a TTY, or video terminal. No one with a serious computer application would consider attempting to enter data from a switch and status light console. These may be educational, but they sure aren't practical. Calculator keypads and digital readouts are not much better. There is no substitute for a full alphanumeric keyboard and terminal system display for serious work.

Mikbug® is a registered trademark of Motorola Inc.

SWTPC 6800

Computer System with serial interface and 2,048 Words of memory $450.00



☐ Enclosed is $450 for my SwTPC Computer Kit ☐ Send Data

☐ or BAC ______________________ # ______________

☐ or MC ______________________ Ex Date ______________

NAME ______________________

ADDRESS ______________________

CITY ______________ STATE ______________ ZIP ______________

Southwest Technical Products Corp., Box 32040, San Antonio, Texas 78284

In the Queue

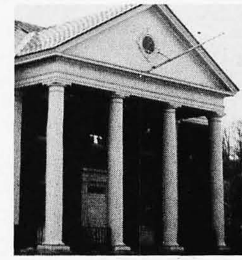
BYTE #6

FEBRUARY 1976

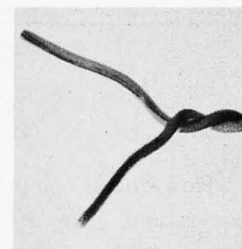
Foreground

Background

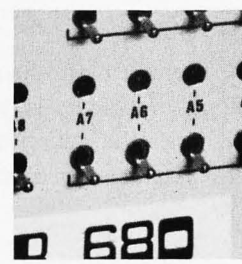
Nucleus



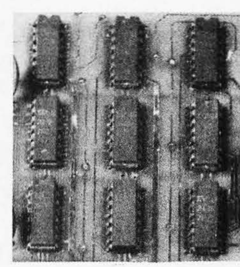
p. 14



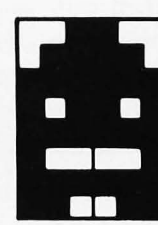
p. 32



p. 42



p. 50



DISPLAY QUEEN

p. 62

BYTE magazine is published monthly by BYTE Publications, Inc., 70 Main St, Peterborough, New Hampshire 03458. Subscription rates are $12 for one year worldwide. Two years, $22. Three years, $30. Second class postage application pending at Peterborough New Hampshire 03458 and at additional mailing offices. Phone 603-924-7217. Address editorial correspondence to Editor, BYTE, 70 Main St, Peterborough NH 03458.

New BYTE phone: 603-924-7217

In This BYTE

How does a computer evaluate a complicated mathematical expression? There are many ways to accomplish this function. One technique is to use the **My Dear Aunt Sally Algorithm** which is described by Robert Grappel.

There is more than one way to **Process Algebraic Expressions,** of course. In his article on the subject, W.D. Maurer describes the Bauer-Samelson algorithm, which uses an operator stack and an operand stack to parse algebraic expressions.

How do bits get from one place to another? The design of **Data Paths** to convey information is an important consideration in any system. Gary Liming provides some background information on the subject.

MITS makes a 6800 product, too. James B. Vice of MITS describes his company's design in an article on **The New Altair 680.**

Hard copy can be created in many ways. Its purpose is to record information in a form which humans can interpret unaided. In **How to Save the Bytes,** Thomas McIntire proposes a multiple-segment character set which might be useful for an inexpensive print mechanism.

Gary Kay continues a presentation of information on the Southwest Technical Products Corporation's 6800 system which was begun in BYTE No. 4. This month's installment presents information about the memory boards, serial interface, control interface, parallel interface, power supply and case. The information is completed with a short discussion of available software and future additions to the system.

BYTE has published a few examples of LEDs used in test probes and diagnostic equipment. In his article E. W. Gray provides some basic information on the use of LEDs. After incorporating his suggestions you can have **LEDs Light Up Your Logic** in more ways than one.

One way to monitor digital data lines is simply to drive an LED based upon the steady state of the line. But what if you are dealing with occasional pulses? A simple indicator is the **TTL Pulse Catcher** described by Bill Walde, a test instrument which can be built from one IC, one LED, two switches and two resistors.

How do you generate graphics patterns for TV raster scanning? One answer is provided by Don Lancaster's discussion of **Color Graphics Techniques.**

Could a Computer Take Over? Ed Rush provides us with some thoughts on the subject with ample references to the speculative fiction of computer technology. We'll let readers draw their own conclusions.

Getting Information from Joysticks and Slide Pots is a problem which must be solved for interactive game purposes. One solution to the problem is shown in this issue.

And on the cover, artist Robert Tinney shows My Dear Aunt Sally at work tutoring a computer on the subject of interpreting arithmetic expressions.

Lowest Price in the World!

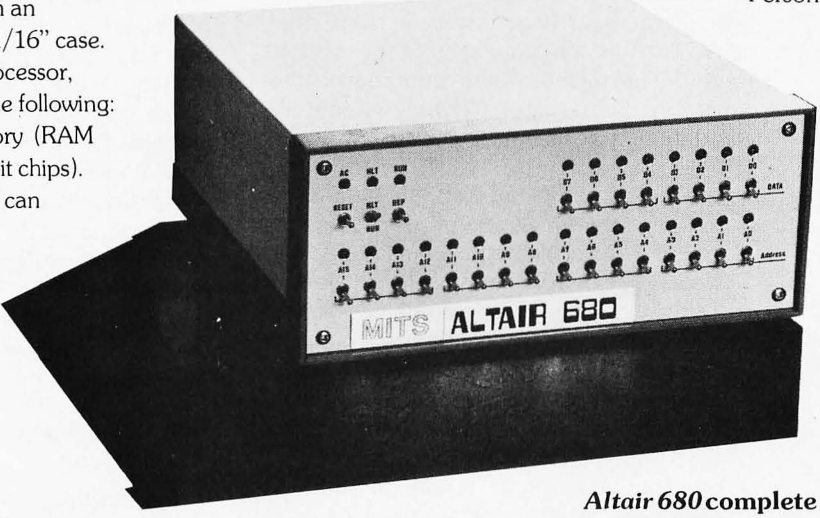
In January of 1975, MITS stunned the computer world with the announcement of the *Altair 8800* Computer that sells for $439 in kit form.

Today MITS is announcing the Altair 680.

The Altair 680, built around the revolutionary new 6800 microprocessor chip, is the lowest priced complete computer on the market. Until December 31, 1975, this computer will be sold in kit form for the amazing introductory price of $293! **(A savings of $52!)**

The Altair 680 comes with power supply, front panel control board, and CPU board inclosed in an 11" wide x 11" deep x 4 11/16" case. In addition to the 6800 processor, the CPU board contains the following:

1. 1024 words of memory (RAM 2102 type 1024 x 1-bit chips).
2. Built-in Interface that can be configured for RS232 or 20 mA Teletype loop or 60 mA Teletype.
3. Provisions for 1024 words of ROM or PROM.



The Altair 680 can be programmed from the front panel switches or it can be connected to a computer terminal (RS232) or a Teletype such as an ASR-33 or surplus five-level Baudott Teletype (under $100).

The Altair 680 can be utilized for many home, commercial or industrial applications or it can be used as a development system for *Altair 680* CPU boards. With a cycle time of 4 microseconds, 16-bit addressing, and the capability of directly addressing 65,000 words of memory and a virtually unlimited number of I/O devices, the *Altair 680* is a very versatile computer!

Altair 680 Software

Software for the *Altair 680* includes a monitor on PROM, assembler, debug, and editor. This software is available to *Altair 680* owners at a nominal cost.

Future software development will be influenced by customer demand and *may include BASIC on ROM.* MITS will sponsor lucrative software contests to encourage the rapid growth of the *Altair 680* software library. Programs in this library will be made available to all *Altair 680* owners at the cost of printing and mailing.

Contact factory for updated information and prices.

Altair Users Group

All *Altair 680* purchasers will receive a free one year membership to the Altair Users Group. This group is the largest of its kind in the world and includes thousands of *Altair 8800* and *680* users.

Members of the Altair Users Group are kept abreast of Altair developments through the monthly publication, ***Computer Notes.***

Altair 680 Documentation

The *Altair 680* kit comes with complete documentation including assembly manual, assembly hints manual, operation manual, and theory manual. Assembled units come with operation and theory manuals. Turnkey model and CPU boards also include documentation.

NOTE: *Altair 680 manuals can be purchased separately. See back page of this catalog for prices.*

Delivery

Personal checks take 2-3 weeks to process while money orders and credit card purchases can be processed in 1-3 days. Delivery should be 30-60 days but this can vary according to order backlog. All orders are handled on a first come, first served basis.

Altair 680 Prices

***Altair 680* complete computer kit $293**
($345 after December 31, 1975)

***Altair 680* assembled and tested $420**

***Altair 680T* turnkey model (complete *Altair 680* except front panel control board) Kit Only $240**
($280 after December 31, 1975)

***Altair 680* CPU board (including pc board, 6800 microprocessor chip, 1024 word memory, 3 way interface and all remaining components except power supply) $180**
($195 after December 31, 1975)

***Altair 680* CPU board assembled and tested $275**

Option I/O socket kit (required when interfacing 680 to external devices) .... $ 29

Option cooling fan (required when expanding 680 internally) .... $ 16
($22 after December 31, 1975)

Option cooling fan installed .... $ 26

PROM kit (256 x 8-bit ultraviolet, erasable 1702 devices) .... $ 42

MITS

"Creative Electronics"

Prices, delivery and specifications subject to change.

Join the Club

Editorial by
Carl Helmers

BYTE staff

PUBLISHERS
Virginia Peschke
Manfred Peschke
EDITOR
Carl T. Helmers, Jr.
GENERAL MANAGER
Manfred Peschke
PRODUCTION MANAGER
Judith Havey
PRODUCTION ASSISTANT
Beth Alpaugh
CIRCULATION
Deborah R. Luhrs
ADVERTISING
Virginia Peschke
TYPOGRAPHY
Custom Marketing Resources, Inc.
Goodway Graphics
PHOTOGRAPHY
Ed Crabtree
Custom Marketing Resources, Inc.
DRAFTING
Bill Morello
PRINTING
Custom Marketing Resources, Inc.
The George Banta Company
ASSOCIATES
Hal Chamberlin
Dan Fylstra
Don Lancaster
Chris Ryland

I found an interesting quotation in the Wall Street Journal in a recent front page news analysis on the subject of philanthropy. The theme of this quotation is the American heritage of forming voluntary associations for specific purposes.

> "Americans of all ages, all stations in life, and all types of disposition are forever forming associations," wrote Alexis de Tocqueville in the 1830s. "Americans combine to give fetes, found seminaries, build churches, distribute books and send missionaries to the antipodes. Hospitals, prisons and schools take shape that way. . . . If (Americans) want to proclaim a truth or propagate some feeling by the encouragement of a great example, they form an association." (*Wall Street Journal*, November 26, 1975.)

This thought provides a timeless introduction to the latest manifestation of that trend, the formation of computer associations and societies for the purposes of advancing the personal use of small systems technology.

To paraphrase an appropriate definition in Webster's, a society is a group of individuals who elect to associate with one another for a common interest or purpose, and who meet periodically to help advance that interest or purpose. Fine. This concept of society could equally well be applied to groups ranging from a regular Saturday night beer drinking circle to a specialized scientific association. How does it apply to our common interest in the technology of small scale computing? Some thoughts on the subject are the substance of this month's editorial.

Information, Friendship and Trade

Defining the purpose of an association is probably the most important task to be accomplished. The beer drinking circle has a purpose assented to by all its members: drinking beer, talking and relaxing at the end of a week's activity. The scientific association has its purpose too: exchanging information, arguing and debating fine points which can lead to productive experiment, and socializing with compatriots in the same field of endeavor.

So what is the purpose of an association of computer experimenters? At the risk of omitting something (to be corrected by reader feedback, no doubt) I've made a tentative list of common purposes which are the defining interests of computer societies. The list contains 3 items:

Information – about computers and their use.

Friendship – fraternizing with other individuals in the field.

Trade – exchanging equipment and learning about what is available.

In making this list, I draw upon examples provided by the similar activities of scientific and engineering societies as well as numerous hobby societies in fields like audio reproduction, amateur radio, radio controlled models, amateur astronomy and model railroading. Not to be omitted of course are the beginnings of similar associations in the small systems field (more on that a bit later).

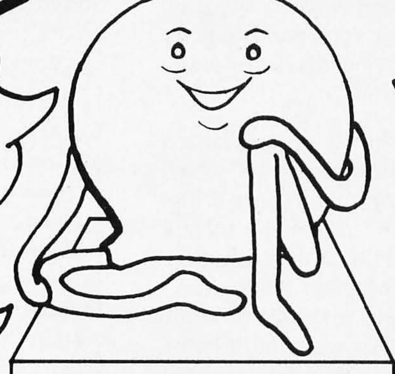
Information

The information activities of a computer society can be one of the most important reasons for establishing these organizations. Information comes in many forms. Given the idea, what are some of the possibilities for specific activities to be arranged by a computer society's education and information committee? I can think of a few and list them, but will make no pretension that this is a complete or final list.

Application reports. Small computer systems can be used for a variety of purposes. One important activity of the computer society is the exchange of ideas and concepts on applications of the machines.

LOOK WHAT WE'VE GOT!

GODBOUT



BILL GODBOUT ELECTRONICS
BOX 2355, OAKLAND AIRPORT, CA 94614

TERMS: Add 50¢ to orders under $10; add shipping where shown. NO CODs! Cal res add tax. Mastercharge® or Bankamericard®: call (415) 357-7007 24 hours a day.

Fast & Easy



FROM VECTOR...slashes time of point-to-point wiring; eliminates cutting to length, stripping, and crimping! 36 ga. wire has special insulation that melts when heated. With 1 red, 1 green bobbin of wire. As reviewed in the December 1975 issue of Popular Electronics. Add shipping for 1 pound.

VECTOR WIRE PENCIL $9.50

OK, all you wire pencil fanatics, we've got replacement wire bobbins. Specify RED, GREEN, BLUE, or CLEAR.

& BOBBINS! 3/$2.40

POWER SUPPLY KITS

± 15 V

A medium current (150 ma per side) supply, designed for economy and stability. Uses a dual tracking regulator; short proof and thermally limited. Perfect for op amp and other applications requiring ±15 volts. $9.95 + shipping for 2 lbs. Less case and hardware.

12 VOLT, 8 AMP

Ideal for powering automotive equipment; also makes a dandy lab supply. Better than .05V regulation, indefinite short protection, adjustable output 11-14 volts, current limits at 13 amps. Kit complete but less chassis and hardware. A BEST SELLER.

$22.50 +7 lbs post.
NEW AND IMPROVED

Easy & Fast



HOBBYWRAP TOOL $41.95

WHY SOLDER YOUR PROTOTYPES?

Wire wrapped connections protect components from heat, are easily modified for corrections or changes, make connections that are better than solder, and can speed up the time required for complex digital projects. If the high cost of wrapping tools puts you off, look at ours; you get the tool (rechargeable so you don't have to deal with trailing cords in tight places), bit, charger, nicads, and instructions.

DUAL TRACKING

This kit provides dual tracking regulated output voltages, both (+) and (-) referenced to ground, from under 1 volt to more than ±18 volts DC. Short circuit protection, thermal limiting, etc. Kit doesn't include case or hardware. Get one for **$10.95** plus 2 lbs shp. Up to 150 ma per side.

FIXED 1/2 AMP

Regulated, short-proof, thermally limited. ½A continuous, or 1 amp intermittent. Specify 5, 8, or 12 volt model. Less case & hrdwre

plus 2 lbs. SHIP

$7.95

SOFTWARE

EROM board kit(s)

So you've got yourself a microcomputer...only to find out you need software to run it efficiently. Give your 8080 editor, assembler, and monitor routines with our 8080 Software Board. **$225** brings you 4kbytes of programmed Eraseable Read-Only Memory, dedicated to making happier 8080s. ...Or maybe you have your own software. Don't load it in your RAM; put it in one of our unprogrammed EROM boards. A 2K x 8 board is $125; a 4K x 8, $200; and 8K x 8, $352. Add 13% to any of the above boards, and we'll program it according to your specifications.

What do all our EROM boards have in common? DIRECT PLUG-IN COMPATIBILITY WITH THE ALTAIR 8800, as well as sockets, industrial quality plated through boards, ultra-low power consumption, buffered inputs and outputs, and lots of bypassing. Includes full instructions and logic print.

5204 $24.50

A 4K EROM IC---similar to the 5203, but twice the capacity. Fully programmable; erase with UV light.

STATIC===NON-VOLATILE===VERSATILE!

2102's $1.95

THE SPECIAL CONTINUES! Guaranteed faster than 750 ns.



CAVE grafix™

We haven't forgotten you RAM fans, either. Our 4K x 8 RAM kit offers DIRECT PLUG-IN COMPATIBILITY WITH THE ALTAIR 8800, as well as the usual high-quality board, bypassing, sockets, buffered addresses and outputs, 500 ns access at 25°C typ. FULL INSTRUCTIONS and logic print included. ENJOY!

4K X 8 RAM KIT

MEMORY

LINEAR SPECIAL

5558 5/$1.95

When a person or informal project group has spent some time developing some aspect of the use of the systems, a report in the form of an informal lecture followed by question and answer sessions is a very good activity for both the person giving the lecture and the rest of the membership.

Show and Tell. This is a specialized version of the application report, in which the society member giving the report brings along some examples of equipment to talk about. Ideas for this kind of activity abound. Do you want to see what an Altair looks like? Have an Altair owner give a demo some meeting. Do you want to see what a home brew microprocessor comes out like? Have the person who has put together one of the contemporary microprocessors show off the result some evening.

The information activities of a computer society can be one of the most important reasons for establishing these organizations.

Elementary Courses Anyone? There is much interest in learning the elements of computer technology. Knowledge of the technology does not grow on trees, and personal interaction with people possessing such knowledge is an excellent way for novices to obtain the required information. One way to disseminate such information is for a computer society to schedule "Novice Nights" devoted to talks on topics in computers chosen to teach new information to people without extensive computer backgrounds. The novice whose primary skills lie outside computer areas is obviously interested in this type of talk and can benefit greatly.

But even the advanced persons in the field can benefit since specialization is so common in these days of high technology. The person with a background in hardware design (the nuts and bolts of computers) is likely to benefit from elementary lectures given by an experienced software person. The software person, on the other hand, can benefit greatly by learning elementary information from the hardware person. Here is an obvious trade of information and ideas between two parties (the third party of the less experienced person can gain from either type of lecture). The act of preparing such talks, whether formally or informally, requires thought on the part of the lecturer in order to review and organize material. This in itself is useful activity and benefits the speaker by adding to his or her skills and experience.

One benefit of bringing the novice up to speed in the intellectual activities of the computer world is that it enables all practitioners in the field to take advantage of the greatest multiprocessor system of them all, a civilization of active, educated and goal-seeking individuals. While these benefits are not immediate, there is long term value in getting people to know the technology. It helps increase the potential for innovation and invention as more and more people learn to use, manipulate and create using the tools of computing.

Technology Updates. Sessions of interest to many people are those devoted to information on new computer technology. Such lectures combine tutorial information of interest to novices with advanced information on the new ways of doing old problems. For example, in a lecture on one of the new bipolar microcontroller chips, it is almost certain that the advanced person would learn about the system architectures of conceivable machines and the types of instructions sets that could be implemented. The novice at the same time would probably pick up some notion of what a microprogrammed machine is, and what it takes to make one.

Special Interest Groups. Whenever two or more people share a common goal which is a subset of the whole small systems computer world, there is potential for a special interest group to be formed. The reason for such a group could be a particular type of processor and its techniques of use, it could be a particular application area such as model trains, radio controlled models, music, amateur radio networks, etc., and it could even be an interest in a side issue such as computer science fiction stories. A computer society can provide the coordination needed to bring such special interests together, by making announcements at meetings, by putting information in newsletters, etc.

Workshop sessions, question and answer sessions and other interactive learning situations. There is no reason that a learning session has to be a one way street. Workshops (the computer society equivalent of an undergraduate laboratory course) are a way to intensively concentrate upon one aspect of the small systems field. This could include programming workshops for learning software principles, and hardware implementation workshops concentrating upon fabrication and testing of elements of the computer systems. One of the most useful activities which a computer society could consider as a regular meeting feature would be the question and answer session. Such a procedure can be very productive. Simply appoint one person as a moderator, who will then search the meeting room for raised hands indicating a query. After restating the question over the PA system (if any) the moderator would then ask for a volunteer to provide information on the question. If the local computer society is full of bashful persons, the questions could be handled in

written form but it would be better all around to use oral methods.

Prerecorded lectures and seminars. Every time a computer society has a talk on a subject of any sort, a good policy would be to record the talk and have several people take notes on any visual materials. It will then be possible for the lecture to be replayed for later use by individuals who missed the first presentation. Such procedure also lends itself to exchanges between computer societies and could become a major activity of a national personal computing society. This kind of policy would be particularly beneficial to local groups in areas remote from such current hotbeds of activity as Los Angeles, San Francisco, Denver and Boston. (More on computer associations at a national level appears a little later in this editorial.)

Friendship

One of the reasons which make special interest societies form and flourish is the common human need to associate with people of similar interests. True computer hackers are noted for being slightly strange, above average in intelligence, and in general somewhere off on the slopes of some multidimensional probability distribution surface of human characteristics. This eccentricity does not mean an absence of the same needs to interact and talk about common interests which are characteristic of humans in general. In fact, the social aspects of the neighborhood computer group are very important because who else would listen and converse with as much enthusiasm as another hacker?

True computer hackers are noted for being slightly strange, above average in intelligence, and in general somewhere off on the slopes of some multidimensional probability distribution surface of human characteristics.

The computer society thus serves a useful function as a framework in which to meet people on a basis of common interest. This immediately establishes a more friendly atmosphere and helps minimize the usual psychological barriers to making new acquaintances. (You don't go out on the street and accost the nearest person to talk about your favorite CPU chip; if you do, you might run certain risks in today's state of civilization.) The selection function which is the computer society's common interest definition guarantees that you have a high probability of a meeting of the minds. Sharing the common interest helps loosen up conversation and create new acquaintances.

Trade

While it is often fashionable in conventional intellectual circles to look down upon commerce and trade with a petty moralistic disdain, the value of trade and commerce is obvious to any person who uses the results of modern technology. Trade and commerce can enter into the computer society in several ways.

Perhaps the most fundamental way in which trade and commerce become an activity of the computer society is in the traditional "flea market" or "garage sale" sessions of such technological social gatherings. If I wanted to unload a spare gate or two, the place to do so would be at a meeting of fellow experimenters. The way it is done is this: each person who has some spare or surplus personal equipment brings that equipment and engages in some old fashioned person to person bargaining in the course of looking for a buyer. (By the way, a computer group can and should generate part of its financial inputs by asking for a commission upon such flea market transactions. A commission of several percent would probably be appropriate.)

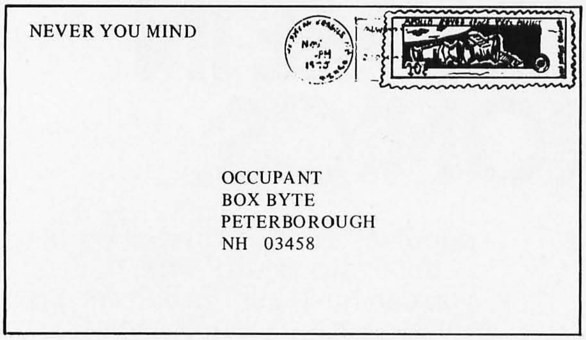
Trade can be fostered at computer society convocations by making arrangements for manufacturers' representatives to come with short talks and demonstrations of their products. As long as such demonstrations are carried out in a spirit of information gathering and "show and tell" nothing but good can come of it. By providing this sort of platform, society members can get a first hand look at the wonderful new gadgets and kits manufacturers supply for them.

The computer society idea can also be used to achieve a price break for members on equipment which is expensive in single unit quantities. Virtually any manufacturer will provide a quantity discount at certain levels of volume. The societies take advantage of this by pooling orders for expensive equipment (two examples are floppy disks and hard copy terminals) then purchasing the items with the usual OEM (original equipment manufacturer) discounts for quantity purchasing. This same concept can apply to a number of items which are virtually unavailable through other channels due to price or a manufacturer's disinterest in servicing small orders. Such activity also can finance a computer society's activities through a percentage fee (SCCS asks 2% on their group purchases). The concept does not have to apply solely to large orders for exotic equipment. It can also be applied to certain parts and printed circuits which can be ordered by a local group in sufficient quantity to justify the price breaks.

Toward a National Personal Computing Society?

All the reasons which make a computer society a useful item to have around one's

Continued on page 95



EVEN CATS LIKE BYTE

Last week my cat tore up my copy of BYTE Number 3, the November issue. I had not been able to finish reading it before this happened and I desperately need a new copy of this issue.

Richard Straedey
Madison WI

ON FILLING VACUUMS

. . . I am 100% in agreement with your editorial in the December issue, concerning "what this country needs," and certainly agree with Mr. Ryland that there is a BIG vacuum in the software area! If the microprocessor/minicomputer companies can't get together on a fairly universal language and instruction set they will never be more than a toy. (By the way, PLEASE – unless you are serious – delete the reference to the "world's greatest toy," "ultimate toys" and stay with "the small systems journal!") A point for reference would be the railroad systems – until they compromised on a *standard gauge* they were not a viable entrant into the transportation marketplace. I feel the same is true of the small systems manufacturers.

Our particular application, for which we have not been able to get a satisfactory solution from the "majors," is accounting, inventory, billing (from time and charges entered manually and by either punch paper or cassette from other automatic terminal equipment), payroll, payables and possibly some other information concerned in a mobile telephone sales and service organization, complete with personal paging, and telephone answering services. We also need to retrieve short messages which we may have stored for a T.A.S. or telephone customer. The majors have a program for portions, but not for nearly all of it, and with my very limited knowledge of the subject I cannot consider our own program! One of the lesser known majors had a demo system up and running at a recent international convention – aimed at our market – and was SWAMPED with interested prospects. I am surprised that someone else does not come forth in the same area with a better system (theirs was very basic!), since the interest and market is there. I would be most interested in a possible assist to someone working in this area at this time.

Keep up the journal – some of it is too deep, but all is interesting!

Duane L Williams
Custom Radio Inc
Caspar WY

BASIC ISSUES

BYTE is fine. I am enjoying it – and learning a few things. Just received the December issue. I must take issue with your editorial in the December issue, however, ragarding your discussion of shortcomings of BASIC.

It would seem to me that your knowledge of BASIC is, perhaps, somewhat out of date or else you have some very restricted "definition" in your mind of what constitutes BASIC. In particular, I refer you to the HP 3000 BASIC Interpreter Manual.

In regard to argument and parameter linkages as stated in your editorial, the HP 3000 BASIC with ten deep nested GOSUBs doesn't seem to me to be "too" primitive; but beyond that the HP 3000 BASIC's multi-branch GOSUB (i.e., GOSUB integer expression OF statement label, statement label, statement label . . .) hardly seems primitive at all, with essentially no limit to the number of branches.

In addition, the defined functions, where both single line and multi-line forms exist (see section VI of the HP 3000 BASIC manual) provide, I believe, all of the GOSUB

capability and more without (I think) the nesting limit since functions can be "called" within a function.

With regard to parameter passing, refer to the HP 3000 BASIC manual pages 6-10 to 6-12, passing parameters by reference or value. With regard to the IF-THEN-ELSE and DO-END constructs, see the HP 3000 manual section on conditional statements beginning at page 2-25. We have there:

a. IF expression THEN statement label
b. IF expression THEN statement
c. IF expression THEN DO statement list ... DOEND

and further,

"The IF...THEN statements can be followed by an ELSE statement to specify action in case the value of the expression is false. Like THEN, ELSE can be followed by a statement, a statement label, or a series of statements enclosed by DO ... DOEND."

My apologies for not sending along a few pages from my HP 3000 Manual. At any rate, I think your editorial does not present an accurate picture of today's [HP 3000] BASIC and its capabilities, and would suggest that you rectify this in a future editorial after looking into it.

In my opinion, BASIC is *far and away* the best high level language around for most purposes, and particularly for engineering, statistical or mathematical purposes.

So that you know where I'm "coming from," let me say that I have no connection whatsoever with HP, other than owning an HP calculator and having occasional access to an HP 3000 system at work.

Thanks for your "ear" and for a good job with BYTE.

Robert E Booth
Santa Clara CA

You make good points, Robert, but omit one important fact: an extended BASIC such as HP 3000 BASIC is no longer the portable language shared in common by several implementations. If you want to write BASIC programs which can be run on any (or most) BASIC interpreters, you must restrict your programs to those BASIC features found as a common denominator of all your intended BASIC implementations. A standard is a standard. For example, the number of changes required to transport an ANSII standard FORTRAN IV program from one manufacturer's implementation of FORTRAN IV to another is very small (but not zero) provided programming is kept to that standard. If you use the (highly desirable) extensions of HP 3000 BASIC, then you limit the use of your programs to HP 3000 BASIC unless extensive modifications are allowed in the process of transporting programs from one machine to another.

However, your enthusiasm for BASIC does prove one important point — any high level language is better than none at all. As tools to aid in design and debugging of software and algorithms, languages like BASIC are an immense improvement over the use of lower level programming techniques.

720 INFO WANTED

This is an out and out letter of praise. I have waited a few months so as not to contribute to the deluge of mail you must have received at the start, and also in case it is slacking off, to let you know it is still great. What more can I say that has not been said before? Perhaps only that BYTE was just what *I* needed at this time.

Quite often when someone gets involved in a subject of great interest he feels that he is the only one in the world interested in it. BYTE has shown that this is not true in personal computers.

I attended MITS ALTAIR caravan seminar the other day and found an excited, involved mob of computer hobbyists who are as serious about this as I am.

I found an excited, involved mob of computer hobbyists who are as serious about this as I am.

While my only interest is in the programming and use of computers in problem solving, I applaud your excellent mix of articles that approach the computer from all sides. Keep the variety going, don't become a software journal, or a hardware tech mag or design sheet. We *all* need to see what the other guys are doing.

Your timing on the keyboard article in BYTE Number 1 was perfect! I had just got one and you really laid it out for me. *Thank you!* One request, how about an article on the 720 (Sanders) terminal. What it is, how it works, how to interface it with a computer, some service info? There must be tens of thousands of them around and their owners would love you for it.

Tom Pappan
Corunna IL

SOME THOUGHTS ON STANDARDS

Let me take this opportunity to congratulate you and your staff for an excellent magazine. This is one publication I read from cover to cover. Keep up the good work!

It's interesting to note the various calls for standardization in the hobby computer field; I heartily agree that there should be as much as possible.

Along this line, let me suggest one more

The computer was designed as a scientific tool; we all don't want to play "super space electronic hangman life-war pong."

item in software which could use some help – the IO problem. Granted, the hardware manufacturers are helping, by offering somewhat standard cards, but what of the fellow who wants to use something other than a Model 33 TTY or a TV Typewriter? For example, I use a parallel-input 30 cps printer, a parallel-output 600 cps tape reader, and a parallel-input punch. I have to write my own drivers for these devices and they can be a lot longer than the standard UART routine. Therefore, any ready-made program I want to run on my computer has to be modified so I can use my drivers with it. If I don't have a listing of the program, I don't know where all the IO routines are located, so I have to de-assemble portions of the object tape to find them; this can be a real bear!

What would be really helpful in a case like this would be to have some standard addresses which would then vector the program to the appropriate IO driver. Every time a printer output is necessary, for instance, the program would call location 100 (or whatever). This three byte section of memory would then contain a jump to the actual IO driver. Thus it would be quite easy to modify any program with this feature to run on any computer; you *know* where the branch table is located and can easily insert your own driver addresses. Naturally, each program could contain its own drivers, if you wanted to use them.

It sure would be nice if someone (BYTE?) would make strong recommendations along this line and if the manufacturers would give serious consideration to adopting the recommendations.

Michael M Dodd
Fairfax VA

A standard for complex peripheral devices has not yet been proposed in the small systems context. There is a de facto plug compatibility standard in the Altair 8800 bus of MITS due to second sourcing of memory cards and peripherals from many firms.

An important factor to be considered by potential software vendors is the issue of modularity and customization to weird peripherals. The most generally useful software will be written with machine-specific "black boxes" assumed for IO operations.

A software package might come with a standard serial IO routine. But if the software is documented and supplied with patching instructions, the serial IO black boxes can be replaced by new routines specific to a particular system.

CHEERS

Several years ago while daydreaming in one of my computer science classes, I formulated what I have since called the programmer's cheer. As a systems programmer, I now recite this occasionally to uplift the spirits of my fellow software and hardware specialists. It is also invigorating to insert this in documentation and in the comments of program listings. Maybe BYTE readers and staff members can use this to inspire enthusiasm and spread the good word?

Programmer's Cheer

Shift to the left
Shift to the right

Pop up! Push down!
BYTE, BYTE, BYTE!

Marilyn Boguski
N Olmsted OH

BEYOND GAMES

Enclosed is $12 for my subscription to BYTE. I think the journal is a fine start, certainly a welcome change from the basement mimeo-outputs I've been getting. A few suggestions, if I may. Advertising is nice, and I know it makes the presses go round, but don't be afraid to be objective in your reviews of products, or to knock an advertiser for poor service, etc. Your readers will appreciate it.

Standardization will be important in the microcomputer field. Everyone it seems will go the cassette route for mass storage and data exchange. It is not too early to look at all the possible cassette software packages available and to recommend the one that seems to be most flexible, least speed-dependent, capable of the most BPI, etc. How about a comparison between Lancaster's design and the pulse modulation system in TCH? The sooner one method becomes accepted, the sooner a free exchange of programs between microcomputer users can begin.

In my own work, I use the computer for data storage, retrieval and statistical analysis using APL, FORTRAN, FOCAL, and the MARS (CDC) and TRIAL (Northwestern) data storage and retrieval programs. Articles on scientific applications would be nice. The computer was designed as a scientific tool; we all don't want to play "super space electronic hangman life-war pong."

Paul Kanciruk
Tallahassee FL

For a limited time only, you can own an Altair® 8800 Computer kit with 4,096 words of memory, new Altair multi-port interface, and revolutionary Altair BASIC language software, for just $695. A savings of up to $114!*



Computer. The Altair 8800 is the best-selling general-purpose computer in the world today. It is a parallel 8-bit word/16-bit address computer with an instruction cycle time of 2 microseconds. It was designed for almost unlimited peripheral and memory expansion, using a bus system where all input/output connections merge into a common line. The Altair 8800 is capable of addressing up to 65,536 words (bytes) of memory. *Regularly priced at $439 for a kit and $621 assembled.*

Memory. The Altair 4K Memory Board provides 4,096 words of dynamic random-access-memory for the Altair 8800. Contains memory protect circuitry, and address selection circuitry for any one of 16 starting address locations in increments of 4K. Access time is 200-300 nanoseconds. The entire 4,096 words of memory on the board can be protected by switching to PROTECT. *Regularly priced at $195 for kit and $275 assembled.*

Interface. Your choice—either the *new Altair 88-2SIO serial interface* or the *new Altair 88-4PIO parallel interface.* The serial interface can be ordered with either one or two ports and the parallel interface can be ordered with up to four ports. *Add $24 for an additional 88-2SIO port kit. Add $30 for each additional 88-4PIO port kit.*

Each port of the *new serial interface board* is user-selectable for RS232, TTL, or 20 milliamp current loop (Teletype). The 88-2SIO with two ports can interface two serial I/O devices, each running at a different baud rate and each using a different electrical interconnect. For example, the 88-2SIO could be interfaced to an RS232 CRT terminal running at 9600 baud and a Teletype running at 110 baud. An on-board, crystal-controlled clock allows each port to be set for one of 12 baud rates. *The 88-2SIO is regularly priced at $115 kit and $144 assembled.*

Each port of the *new parallel interface board* provides 16 data lines and four controllable interrupt lines. Each of the data lines can be used as an input or output so that a single port can interface a terminal requiring 8 lines in and 8 lines out. All data lines are TTL compatible. *The 88-4PIO regularly sells for $86 kit and $112 assembled.*

Software. Altair 4K BASIC leaves approximately 725 bytes in a 4K Altair for programming which can be increased by deleting the math functions (SIN, SQR, RND). This powerful BASIC has 16 statements (IF . . . THEN, GOTO, GOSUB, RETURN, FOR, NEXT, READ, INPUT, END, DATA, LET, DIM, REM, RESTORE, PRINT, and STOP) in addition to 4 commands (LIST, RUN, CLEAR, NEW) and 6 functions (RND, SQR, SIN, ABS, INT, TAB, and SGN). Other features include: direct execution of any statement except INPUT: an "@" symbol that deletes a whole line and a "←" that deletes the last character; two-character error code and line number printed when error occurs; Control C which is used to interrupt a program; maximum line number of 65,529; and all results calculated to seven decimal digits of precision. *Altair 4K BASIC is regularly priced at $60 for purchasers of an Altair 8800, 4K of Altair memory, and an Altair I/O board. Please specify paper tape or cassette tape when ordering.*

*** Savings depends upon which interface board you choose. An Altair 4K BASIC language system kit with an 88-2SIO interface regularly sells for $809. With an 88-4PIO interface, this system sells for $780.**

NOTE: Offer expires on March 30, 1976.

MITS

"Creative Electronics"

MITS/6328 Linn N.E., Albuquerque, NM 87108 505/265-7553 or 262-1951

MAIL THIS COUPON TODAY!

☐ Enclosed is check for $________
☐ BankAmericard #________ ☐ or Master Charge #________
☐ Altair BASIC System Special ☐ 4PIO interface ☐ 2SIO interface
☐ Cassette tape ☐ or paper tape
☐ Extra SIO port ☐ ____ Extra 4PIO ports
Add $8 for postage and handling.
☐ Please send free literature

NAME________

ADDRESS________

CITY________ STATE & ZIP________

NOTE: Personal checks take 2-3 weeks for clearance. For immediate processing send money order or use charge card. Delivery: 30 days. Prices, specifications and delivery subject to change.

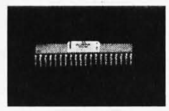
Our New Offices



Publishing has got to be one of the craziest fields of endeavor in the known universe, especially as practiced in up country New England. After a start which reads like a romantic light opera with an episode or two reminiscent of the Keystone Cops, BYTE magazine finally has moved into separate offices of its own.

BYTE is now located at 70 Main Street, Peterborough NH 03458 (phone 603-924-7217). We'll leave the complete story for our memoirs, but the fact is we've moved across town to the American Guernsey Cattle Club building (known as "the Guernsey" to Peterborough residents). We're located on the fourth floor of Peterborough's largest building, a veritable skyscraper if it weren't for Monadnock and Pack Monadnock mountains towering above us nearby. Circulation and billing will soon be changing over to the Guernsey's IBM 360 Model 25 computer (not very exotic but quite functional) and much of our typesetting and graphic work is now being done by the craftspeople at the cattle club. That 360/25 is itself evidence of progress – the Guernsey used to have almost 200 clerks keeping track of the family trees of pure bred cows before the computer automated the records. If you're passing through Peterborough, stop by our humble abode and say hello. ■

You can buy this microcomputer for $39.95, but...



MC 6800

We would be a bit surprised if you could do anything meaningful without additional hardware and software. **Wave Mate's Jupiter II**™ isn't the kind of microcomputer kit you only stare at . . . when you've completed your Jupiter II just plug in your teletype and you're ready to go. That's because it goes beyond the sum of its high quality parts. It's the ultimate micro kit experience. **In performance, in documentation, in reliability.** First, consider its superb features. It has small **pluggable wire wrapped cards** easily tailored to suit your modifications. **Every IC is socketed** and **100% burn-in tested.**

In fact every part including the powerful MC 6800 CPU and the 8K dynamic RAM is guaranteed for 120 days. It has the best software around, **System Monitor** and **Debug programs** (ROM). Includes powerful **text editor** and **Motorola compatible assembler.** And BASIC at no extra cost. Because we've been making microcomputer systems for over 4 years, we can offer you the broadest line of interfaces including TV terminal and dual audio cassette. Impressive. And yet the grandest feature is the experience of completing a kit that works. Guaranteed.

Wave Mate

ACT NOW AND SAVE.
Introductory price $1225. (Assembled $1885.)
(Good until February 1, 1976)

☐ Send Jupiter II Microcomputer Kit. Includes:

- Modular plug-in power supply
- 9 module PC backplane
- CPU module
- System monitor module
- 8K dynamic RAM module
- Serial RS 232 communication interface module
- Front panel module
- Front panel
- Wire; cut, stripped, color coded.
- Rack mount module cage
- Wire wrap tool
- Wire unwrap tool
- Cables, connectors, all other necessary hardware
- Software (editor, debug, assembler, BASIC)
- Assembly manuals
- Operators manuals
- Theory of operation manuals
- Annual membership in users group



☐ Send details on kit-a-month club
☐ Kit ☐ Assembled
☐ RS 232C conversion kit (teletype) $47.85
☐ I'm not convinced; send me 101 reasons why.
☐ Please send free Jupiter II System catalogue.

☐ Check enclosed for $ ____________
Include $10.00 for postage and handling. California residents add 6% tax   Delivery 60 days ARO

☐ Master Charge # ____________
☐ Bank Americard # ____________
Credit card expires on ____________

Name ____________
Address ____________
City/State/Zip ____________

WAVE MATE
1015 West 190th Street Gardena, California 90248
Telephone (213) 329-8941

Wave Mate

WARRANTY
120 days on all parts, materials and workmanship.

Keyboard Modification

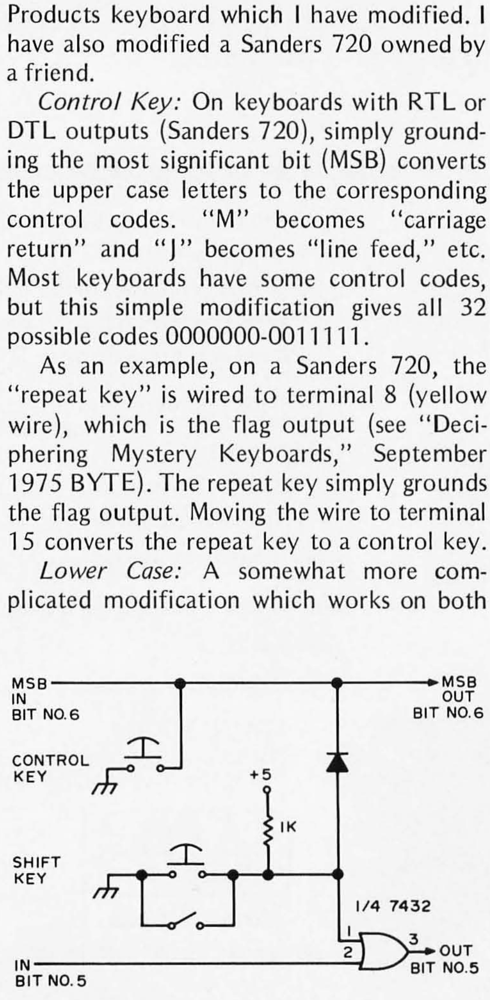
George Macomber
1422-18th Ave
Seattle WA 98122

I read your article in the September 1975 BYTE on surplus keyboards with interest. I have made some simple modifications to produce lower case codes on RTL and DTL keyboards. I have a Southwest Technical Products keyboard which I have modified. I have also modified a Sanders 720 owned by a friend.

Control Key: On keyboards with RTL or DTL outputs (Sanders 720), simply grounding the most significant bit (MSB) converts the upper case letters to the corresponding control codes. "M" becomes "carriage return" and "J" becomes "line feed," etc. Most keyboards have some control codes, but this simple modification gives all 32 possible codes 0000000-0011111.

As an example, on a Sanders 720, the "repeat key" is wired to terminal 8 (yellow wire), which is the flag output (see "Deciphering Mystery Keyboards," September 1975 BYTE). The repeat key simply grounds the flag output. Moving the wire to terminal 15 converts the repeat key to a control key.

Lower Case: A somewhat more complicated modification which works on both the Sanders 720 and the Southwest Technical Products keyboards allows the production of lower case letter codes 1100000-1111111.



In order to get lower case codes from a keyboard which produces only upper case, it is necessary to make the fifth bit high. The code for "A" 1000001 becomes "a" 1100001, and "[" 1011011 becomes "{" 1111011. The circuit shown adds lower case and control to any RTL or DTL output keyboard.

The control key has already been mentioned and is not required if the keyboard already has one (Southwest Technical Products). Both a shift key and a toggle switch are shown. You will probably want both. When the toggle or key switch is closed, the keyboard behaves as it did before modification. When both are open, the keyboard generates lower case, but the numbers and other shifted keys (i.e., 1 → !) are unaffected. A convenient key to use on the Sanders is one of the shift keys, leaving the other shift key for numbers and some other symbols (i.e., [, \ ,], _).

What happens when both shift keys are pressed? Shift' has no effect on the numbers since bit no. 6 is Ø, which forces the upper case or shift' function. But the old shift changes the letter codes, either by forcing the fourth bit to Ø, or by inverting the fourth bit (Sanders). Inverting the fourth bit allows the generation of some additional codes ([, \ ,], _), and their equivalent lower case ({ , } , ≈, DEL) and control codes. These will not be available if your keyboard forces the fourth bit, unless it has separate keys for these codes. ■

The "My Dear Aunt

Robert Grappel
MIT Lincoln Laboratory
Lexington MA 02173

The number of mathematical operations a computer can perform without the aid of programming is quite small. The bare machine can add and subtract, and perhaps it can also multiply and divide. It cannot comprehend a series of operations, nor can it evaluate a mathematical expression as a human would typically write it. It cannot group operations as required by the rules of mathematics. All these require software; programs which convert mathematical statements to sequences of machine instructions. This article describes a set of programs which can read a mathematical statement in its normal form and evaluate its result. The discussion is kept on a general level: no specific machine or language structure is assumed. These programs should find their way into many "do-it-yourself" assemblers, compilers and interpreters .... wherever it is useful to write mathematical expressions for input to a computer.

My Dear Aunt Sally

Shortly after students learn to do the basic operations of addition, subtraction, multiplication, and division (the same operations that bare computers can perform), they are faced with problems of the following kind: 3–5*2=?. A student with imagination finds such a problem paradoxical; there are several apparently "correct" answers. Performing the operations in the order of appearance, he gets –4 (scanning left-to-right). Performing the multiplication first, he gets –7. If he chooses to scan from right to left, then the results +4 and +7 are possible. Which is correct?

My teacher gave us a rule to remember how to proceed with these problems – multiply, divide, add, subtract – or, "My Dear Aunt Sally" as we came to remember this order. Mathematical statements are read from left to right, for each operation. The evaluation starts with all the multiplications, left to right. It then proceeds in the order of "my-dear-aunt-sally," evaluating all the divisions, then all the additions, then all the subtractions. Mathematicians call this ordering the "precedence of functions," and all mathematical operations can be ranked in the order in which they are to be performed. Hence, the example has only one correct result, and this is –7.

Unfortunately, precedence is not enough to force a single answer from every problem. Suppose one wishes to perform one operation upon the result of a group of several other operations, some of which are of higher precedence. One needs some mechanism to group certain parts of a mathematical statement so that they can be considered as a single unit to be treated by some other operation. For this purpose, mathematics uses parentheses. If one wrote the example from above as: (3–5)*2=?, it is clear that the subtraction should be performed first, in spite of the precedence of the two operators. With the two tools of precedence and parentheses, one can force the desired ordering of operations upon a mathematical statement and ensure that there will be only one correct result.

What Is a Parser?

Now, how can a computer deal with complex mathematical statements like the example? The computer can perform each operation individually, either by a single

Sally" Algorithm

elementary operation or by calling a subroutine. The problem is the same one that we faced as students; how does one break down a complex mathematical statement to perform the individual operations in the right order? The name for this operation of breaking down a complex statement into its component parts, determining the structure of the statement, and evaluating it as required, is "parsing." This article describes a set of procedures which together form a parser.

Tokens

At the outset, the computer sees a mathematical statement as a string of characters. All that is known about the string is its starting address and its length. The statement

X–2/(YAXIS+Z)

is a string of 13 characters at some address. One of the first things that is necessary is a procedure to subdivide this string into its elements: variables, constants, operators, and parentheses. The example contains three variables: X, YAXIS, and Z. It contains one constant: 2. There are three operators: –, /, and +. There are also two parentheses. Each of these elements is a character string. These strings may be of differing lengths. There may be blanks embedded in the input string, but these are not desired in the element strings. The procedure which subdivides the input string and eliminates blanks is called NEXTOKEN. Each element of the input string is called a token. The first problem of constructing a parser is to find a way to inform a computer about the tokens contained in the character string representation of a mathematical statement.

Blanks as Separators

There are several ways to approach this problem. Perhaps the easiest (in the sense that the coding of NEXTOKEN is simplest) is to require that the writer of a mathematical statement put at least one blank between every element or token in the statement. In this way, the human programmer breaks the input string into tokens before the computer gets it. We would require that the above statement be written as

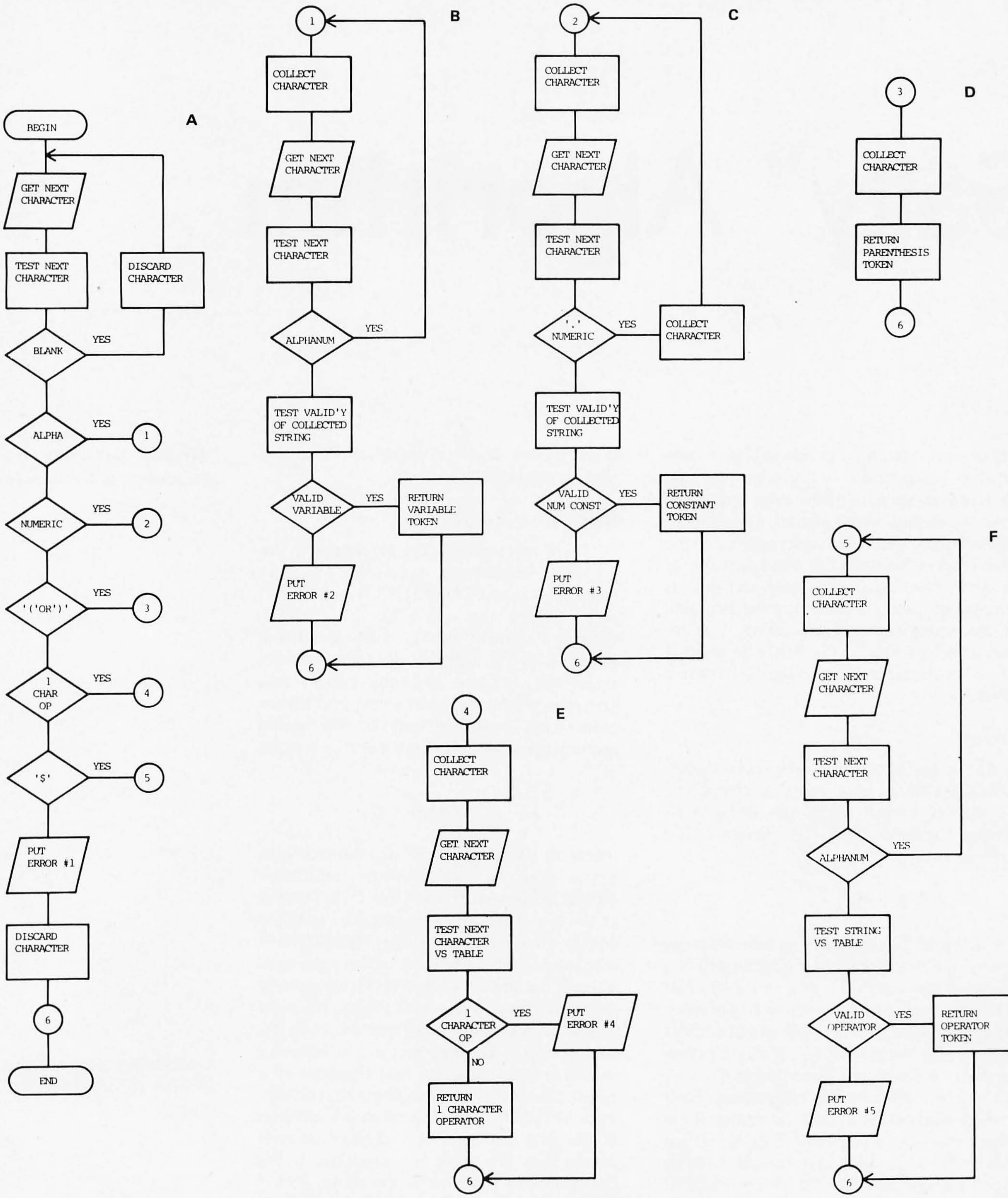
X – 2 / (YAXIS + Z)

where all the extra spaces are required. With this method of token separation NEXTOKEN would work like this: Starting at the last character processed (the leftmost one at the start of the string), NEXTOKEN scans the input string from left to right until a blank occurs. This substring (from starting point to blank) is the next token. The next step is to determine what type of token has been scanned. The rule that will be followed in this article is that the first character of a token determines its type. If the first character is alphabetic, then the token is a variable. If the first character is a digit or decimal point, then the token is a constant. If the first character is neither of the above, then it is checked against a table of legal operators. If it is not an operator, then it is checked to see if it is a parenthesis. Variables might be checked for invalid characters in their names or too many characters in the name; constants might be checked for non-numeric

"My Dear Aunt Sally" is a precedence ordering rule.

A parser is a programming scheme to analyze statements.

Figure 1: Flow chart of the NEXTOKEN algorithm used in this design. (a) The main routine. (b) The variable name collection algorithm. (c) The numeric constant collection algorithm. (d) Parenthesis handler. (e) Single character function name handler. (f) Generalized function name handler.



A character is fetched by the "get next character" operation. This character may be collected into an output string by NEXTOKEN, or discarded if it is to be ignored. If it is neither collected or discarded, the character will be reused the next time the "get next character" operation is performed. Flow is from left to right or from top to bottom unless an arrowhead indicates a different flow direction.

characters or more than one decimal point. A token which failed to match the model of any of the four token types would be flagged as an error, as would any of the error conditions described above. A check for string length of an operator might be used: The string '+=VAR' is not an operator, despite its first character.

A Smarter Token Separator

Requiring blanks around every token of a mathematical statement not only takes up valuable memory space but also makes the parser very susceptible to programmer errors. It is far too easy to forget one of those critical blanks. Fortunately, with a slightly more complicated mechanism for NEXTOKEN, one can parse a randomly written statement with any spacing. This algorithm is flowcharted in figure 1.

The routine starts by always scanning and ignoring any leading blanks. Eventually it finds a non blank character, the first character of some token. Remember that a rule was established for this parser: The first character of a token determines its type. Once the token type is known, the NEXTOKEN routine checks that subsequent characters are valid for that token type. As soon as a character is found which is invalid for that token type, the token is completed. For example, consider the expression X+5. The first character (leftmost) is alphabetic; this means that the first token is a variable. The second character is not alphanumeric, so it is not part of the variable type token. Hence, the first token is X which is a variable. The next token starts with a + which is found in the operator table. Hence it is an operator. The next character is the digit 5. This is the start of a constant type token. There are no further characters, so the token is complete.

Is It an Operator or a Variable?

One useful extension to the algorithm should be considered here. Many operator names that we would like to use are not single symbols, but are several characters in length. One might for example want to call the cosine operator by the name COS. Unfortunately, the simple minded NEXTOKEN procedure would confuse this with a variable named COS. There are several ways around this problem. One is to define new symbols for each operation added to the system. These are added to the character tests for operators and the parser will work fine. This makes for clumsy notation, however; and there may not be enough distinct characters available in keyboards, Teletypes, etc. A second approach is to require a special character, such as the dollar sign ($), as the first character of the string desired as an operator name. The cosine function might therefore be named $COS. The dollar sign would disqualify the name as a variable and identify it as a candidate for operator status. NEXTOKEN would then check the remaining characters in the name (using the same rules as for variables) against a table of operator names. If a match is found, then the string is an operator. The third approach is to forego the identifying first character in operator names and to treat operator names as a special kind of variable in NEXTOKEN: When a character string is typed as a variable, it is then checked against the list of operator names. If a match is found, then the token is changed in type from variable to operator after further statement analysis. Since this is complicated, NEXTOKEN assumes the dollar sign as identifying character for extended operator names. We see that NEXTOKEN starts with a character string, a starting pointer within that character string, and the position of the end of the string. NEXTOKEN returns a character string which is the new token, an indication of the token type, and the token length. It leaves the input string starting pointer with a new value after collecting or discarding each character needed to build the current token.

Some Small Procedures

There are several small procedures which are necessary to the parser and which are briefly described now. Two of these are required to convert the character strings, which are the tokens, into their values. One of these, called CONST, works on constants. The other, called VARIABLE, works on variables. Routines like these are usually available in a large computer's operating system. For minicomputers, the algorithms can be extracted from programming texts and programmed for software deficient home brew systems. The mechanism of VARIABLE depends on the structure chosen for the symbol table used to store variables. There should be some form of check that a variable has a value before it is used. If it has no value, an error message should be generated. Another routine needed is a form of branch table to convert the character string name of an operator into a call to the proper subroutine to perform the operation.

A mechanism for generating the precedence of operators is also needed, as was demonstrated in the introduction to this article. This amounts to a table of precedence values indexed by the operator name. Every legal operator is assigned a prece-

Computing the value of a statement is often easier if the statement is first rewritten in a form better suited to computers. Polish notation is such a form.

Operator	Precedence
Trigonometric functions, Logarithmic functions, Roots, etc.	7 or more
↑ (Exponentiation)	6
* (Multiplication)	5
/ (Division)	4
+ (Addition)	3
— (Subtraction)	2
parenthesis	1
empty stack	0

Table 1: Precedence of operators. This is a typical mathematical precedence of operations.

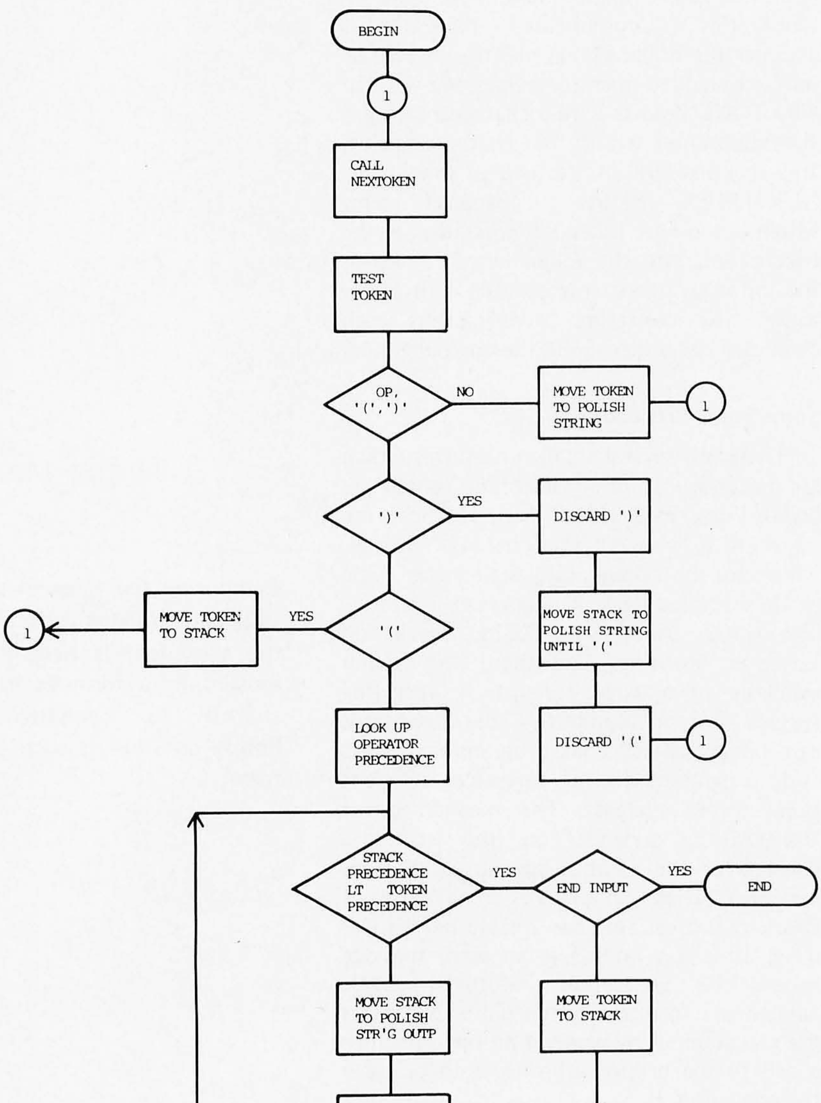


Figure 2: The POLISH routine specified as a flow chart. This routine uses NEXTOKEN to obtain individual tokens of the syntax, then uses its logic to rearrange these tokens in Polish notation. The output of this routine is a Polish string.

dence, as are parentheses and the right and left terminations of the input string. This article assumes that the higher the precedence value, the earlier the operation should be done. In other words, the operator with highest precedence goes first, the second highest goes next, etc. Table 1 is a typical precedence table. This precedence lookup procedure is called PRECEDE.

Intermediate Results

As shown in the introduction of this article, it is often necessary to evaluate a mathematical statement in a different order than that in which the statement was written. "My Dear Aunt Sally" does all the multiplications first, then the divisions, then the additions and subtractions. This parser operates by rewriting the statement into a more desirable form. In the new form, the operators are performed in the order that they appear. There are no parentheses required in the new form. The operands (the variables and constants) and operators are located in a simple and consistent relationship to one another in the new form. This makes the evaluation algorithm easier to write and more efficient. This "nice" form for writing mathematical statements was devised by the Polish logician J. Lukasiewicz. The next item to consider in this parsing system is a procedure to rewrite the input character string into Polish notation. This procedure, called POLISH, uses NEXTOKEN to disassemble the input string when required by its algorithm. POLISH also uses PRECEDE to compare the precedences of operators in different parts of the string. Figure 2 shows a flowchart of POLISH. This procedure forms the heart of the parser and is described in some detail. Several examples are also worked through. Once a mathematical statement is in Polish notation, it is rather easy to evaluate it.

A few examples of Polish notation and its evaluation are in order here. Consider the statement A*B+C*D. We know by precedence that the multiplications are to be performed first. In Polish notation we choose to write the operands first, followed by the operator. Hence, A*B becomes AB*. Similarly, C*D becomes CD*. Then the addition of these quantities is written AB*CD*+. This is the Polish form of the example statement. This string is evaluated as follows: Starting at the left, operands (variables and constants) are passed directly into a software pushdown stack in memory, which is a temporary holding area organized so that the first item in the stack will be the last item out of the stack. (This stack should not be confused with the subroutine call and return stacks of many microcomputers.) The

stack will hold the *values* of the operands, as returned by the procedures VARIABLE and CONST. When an operator is encountered, it is applied to the top two values in the stack. The result of the operation is returned to the stack in place of the operands. Figure 3 shows a flowchart of the evaluation procedure, called EVAL.

Let us proceed to evaluate the example AB*CD*+. First, we place A, which is a value, on the stack. Then we place B on the stack. Next, we encounter the operator *. The top two values on the stack are A and B, so we compute A times B and return that value to the stack. Next, we put C on the stack. Next, D goes on the stack. Then the operator * is encountered again. The top two entries on the stack (the last ones entered) are C and D. We compute C times D and return that value to the stack. The stack now holds two values which are the products of A and B, C and D, respectively. Finally, we encounter the operator +. We perform the addition and then we are done. In a similar manner, the first example of this article, 3–5*2, would be written as 352*– in Polish.

POLISH starts by calling NEXTOKEN for the first token. If it is not an operator or parenthesis, the token is output to the Polish string and NEXTOKEN is called for the next token in the input string. If the token was a left parenthesis, the parenthesis is placed in the stack and NEXTOKEN is called again. If the token was a right parenthesis, the contents of the stack are moved to the output Polish string until a left parenthesis is encountered or the stack is empty. Both the left and right parentheses are deleted and NEXTOKEN is called. Parentheses must occur in left-right pairs – if there is no left parenthesis in the stack after a right parenthesis is found, there is an error and the string cannot be parsed. If the token was an operator, then its precedence is checked against the precedence of the top of the stack. If the new operator is of lower precedence than the top of the stack, the top of the stack is output to the Polish string and the check is performed with the new top of stack. Eventually the new operator will have higher precedence than the top of the stack (an empty stack has zero precedence). If the new token is the end of the input string, then it is treated as an operator of lowest precedence. Some languages use a special character for the input string terminator, but this is not necessary. In any case, if the new token is the end of the input string, then POLISH is finished when the stack is empty. If the new token is not the end of the input string, then the token is placed on the stack and NEXTOKEN is

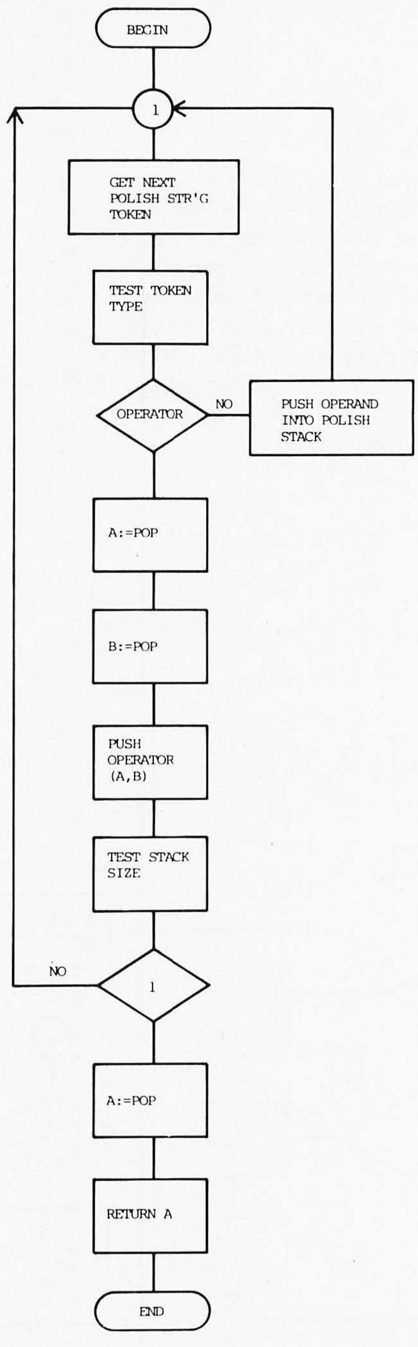


Figure 3: The EVAL routine specified as a flow chart. This routine is an example of an interpreter. It takes the Polish string created by POLISH, and decodes it and evaluates the mathematical value to be computed. Several functions are employed by the EVAL routine, as follows: PUSH means place the value in question into the operand stack, increasing the stack size by one value. POP means recover the top operand from the operand stack, decreasing the stack size by one value. OPERATOR(A,B) means evaluate the combination of the value A and the value B according to the definition of the current operator in the POLISH string. The data concepts employed during evaluation are as follows: Temporary data storage is found in A and B. The Polish string is a series of separated tokens created by POLISH as its output. The operand stack is a first-in-first-out stack of values defined by operand tokens (variables and constants) in the Polish string, or by the results of previous operations which are left in the stack.

Table 2: *An example of the POLISH routine in operation. The output string of a practical implementation of POLISH must have a convention to separate tokens. Storing output as a reconstructed character string with blanks to separate operands allows EVAL and UNPOL to use the same NEXTOKEN routine which POLISH calls. Other storage techniques, which do not require the use of blanks as separators, are possible.*

Input	Stack	Output
A		A
+	+	
B	+	B
↑	+↑	
(	+↑(	
C	+↑(	C
*	+↑(*	
D	+↑(*	D
/	+↑(/	*
(	+↑(/(	
D	+↑(/(	D
+	+↑(/(+	
F	+↑(/(+	F
)	+↑(/	+
*	+↑(*	/
G	+↑(*	G
)	+↑	*
end of string		↑
		+

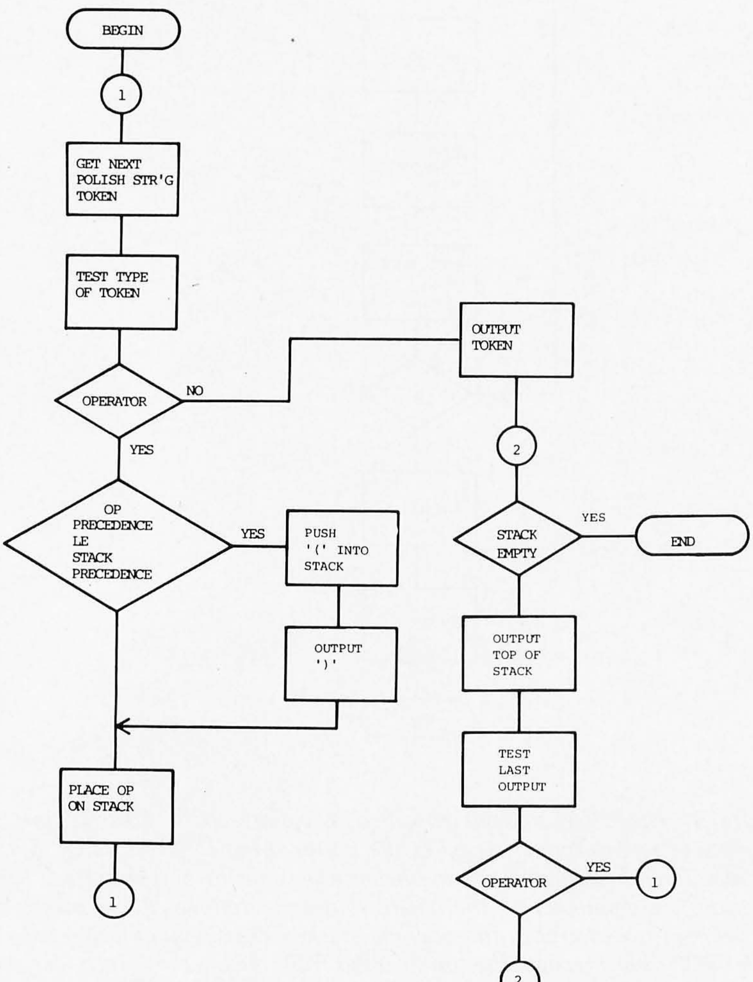


Figure 4: The UNPOL routine specified as a flow chart. This routine takes the Polish string created by POLISH and inverts the transformation to obtain a normal arithmetic expression again.

called again. Table 2 shows the input string, stack contents, and output string as POLISH works through the string A+B↑(C*D/(D+F)*G) where the upward arrow symbol represents the exponentiation. Exponentiation has a higher precedence than the other operations in this example. Working through the example shown in table 2 should convince the beginning programmer that this algorithm actually does translate to Polish notation. EVAL can then evaluate the Polish expression to obtain the final result.

Undoing What's Just Been Done

Polish notation is a convenient way to store a mathematical expression in computer memory. It tends to contain fewer characters, since parentheses are not needed. Also, it can be readily evaluated without the need to first perform a complicated conversion of the sort we just saw described. However, if one wants to edit an expression or change its structure, then one would really like to see the original form of the expression. Figure 4 shows the flowchart of a procedure called UNPOL which reverses the process of POLISH and converts a statement of Polish notation back to normal form. It scans from right to left (the reverse of POLISH) and outputs the normal string in reverse order. UNPOL can use the same NEXTOKEN and PRECEDE that POLISH uses (see the note in table 2). The only change is the sequence in which the tokens are used. Table 3 shows the input, stack contents, and output of UNPOL as it reverses the processing of the example A+B↑(C*D/(D+F)*G). Note that unnecessary parentheses are dropped when a mathematical expression goes through POLISH and then through UNPOL. For

Input	Stack	Output
+	+	
↑	+↑	
*	+↑(*	)
G	+↑(	G*
/	+↑(/	
+	+↑(/(+	)
F	+↑(/(	F+
D	+↑(	D(/
*	+↑(*	
D	+↑(	D*
C	+	C(↑
B		B+
A		A

NOTE: Processing starts at the top of this table. The Polish string is scanned in reverse order starting with its rightmost character. Proceeding down the table, the output is generated in reverse order also, starting with the rightmost character.

Table 3: An example of UNPOL in operation. The Polish string input to UNPOL is scanned in reverse order (right to left) and generates the output string starting at the left.

example, (A*B)+(C*D) becomes A*B+C*D. The parentheses were unnecessary because operator precedence ensured that the multiplications would be done first. UNPOL will not drop any necessary parentheses.

Trying a few examples through the parsing algorithms presented here should convince even a beginning programmer that Polish notation provides a straightforward way to make a computer evaluate complex mathematical expressions. Using these algorithms, it will be possible for readers to incorporate evaluation of mathematical statements into their programming systems.■

My Dear Aunt Sally's Glossary

Alphabetic Character: Any of the letters A through Z.

Assembler: A program which translates symbolic assembly language intput into machine language output. Assemblers frequently require arithmetic statement parsers in order to compute addresses and data values based upon symbolic assembly language statements.

Compiler: A program which translates symbolic statements of a high level language input into a machine language output. Compilers require some form of arithmetic statement parsing, although the output is generally converted one step further into actual machine code.

Constant: A constant is a way of specifying data which is fixed. In the My Dear Aunt Sally parser, constants are defined by input character strings which begin with a numeric character, and contain only numeric characters or at most one decimal point.

Interpreter: A program which translates symbolic statements of a high level language input into an immediate action. An interpreter could use the My Dear Aunt Sally parser to evaluate arithmetic expressions when required.

Mathematical expression: An input character string which obeys the syntactical rules of the My Dear Aunt Sally parser and can potentially be evaluated as a single resulting arithmetic value.

Numeric character: Any of the numbers 0 through 9.

Operator: An operator is a token specifying an action to be taken when the expression being parsed is evaluated. My Dear Aunt Sally recognizes two kinds of operators: Single character operators are used to denote the conventional arithmetic operations; multiple character operators are recognized by a dollar sign (as in $SIN) and are used for mathematical functions.

Parenthesis: Left and right parentheses are used to group operations in mathematical expressions. The only requirement for consistent evaluation of expressions is that left and right parentheses must balance.

Parser: A computer program mechanism for performing the parsing function.

Parsing: Given a set of syntax rules (a grammar) and an input string, parsing is the process of breaking that input string into a series of tokens according to the syntax rules.

Pop: Remove an element from a stack storage mechanism, in a last in, first out order.

Precedence: In evaluating an arithmetic expression, precedence is used to resolve ambiguities in the order of execution of several operations: The operations with higher precedence are performed first.

Push: Add an element to a stack storage mechanism.

Scan a string: The process of sequentially looking at each character or token of a string in a well defined order from left to right, or right to left.

Software pushdown stack: A stack storage mechanism can be implemented exclusively in hardware, or by using mechanisms which are part hardware and part software, or entirely in software. For the purposes of the My Dear Aunt Sally algorithm, all stacks are implemented in software. This means that each stack reserves a random access memory region and is supported by subroutines to perform the push and pop functions. The POLISH routine uses an *operator stack* to temporarily store and reorder the operator tokens when creating a Polish string; the EVAL routine uses an *operand stack* to temporarily hold values as it interprets the Polish string.

Statement: A statement is the programming language equivalent of a sentence in a natural language such as English.

String: A string is a series of values with definite starting and ending points. The parser of this article requires an input *character string* containing the human readable codes of an arithmetic expression, and produces a *Polish string* output of parsed tokens which can be evaluated by the Polish string interpreter.

Subroutine: A subroutine is a section of a program which is *called* to perform its function. When completed, it *returns control* to the routine which calls it. Subroutines are created for two purposes when programming: To modularize a program according to function, and to share common functions and save memory space.

Symbol table: A central collection of the variable names used in a program, along with related information. For the My Dear Aunt Sally parser, a symbol table would be composed of the variable token (a character string) and current numeric value for each variable found while parsing a statement. Note that the My Dear Aunt Sally algorithm by itself does not provide a means for setting the value of variables; an extension of the software into a full interpretive high level language with an assignment statement would provide such a means.

Token: A token is a basic unit of the syntax of an expression. In the My Dear Aunt Sally parser, tokens are character strings collected and returned by NEXTOKEN along with an indication of syntactical type.

Variable: A variable is a symbolically named data location. The parser of this article detects variables as character string names which begin with an alphabetic character.

You don't need high-powered compiler theory to process your own algebraic expressions – all you need are a few variations on one basic idea, developed in West Germany . . .

Processing Algebraic Expressions

W Douglas Maurer
University Library Room 634
George Washington University
Washington DC 20052

To the amateur programmer, algebraic expression processing may seem a formidable obstacle. How do you write a program which takes a character string like 2+3*(4–(14/7–1)) as input, and produces the right answer – in this case 11 – as output? The programmer seeking answers to such questions is usually led to a collection of sources on compiler theory, and to arcane-sounding terms like "recursive descent," "top-down and bottom-up parsing," and the like. These were developed for use by the compiler writer, although even compiler writers find much of compiler theory interesting for theoretical purposes only. The net result has been, in all too many instances, to scare the ordinary programmer away from algebraic expressions entirely – a decidedly unfortunate state of affairs.

Most people who do work with algebraic expressions in a small system setting have made use of what is called "Polish notation." Although we shall describe Polish notation next month, the warning must be given that Polish notation can be misused as easily as it can be used. The much more direct method which we shall describe was developed by F.L. Bauer and K. Samelson at the Technische Hochschule in Munich, Germany. We refer to it as the "Bauer-Samelson algorithm."

Before describing the Bauer-Samelson algorithm, let us first take up a number of elementary points about the processing of algebraic expressions. The input to any algebraic expression processor will, of course, be a string of characters. These are given in some sort of character-code format, and there are as many such formats as there are computers. Even the number of bits per character varies from one system to another. Some systems use five bits per character, some six, some seven, but most use eight – the standard IBM 360 (and 370) "byte."

Since there are 26 letters in the alphabet, at least 26 different codes must be used. To find out how many bits are needed to represent that many codes, we take the next higher power of 2, in this case 32, or 2^5. There are 32 different possible codes in 5 bits (from 00000 to 11111). Therefore 5 bits are enough to represent the 26 letters of the alphabet; whereas 4 bits would not be, because there are only 16 possible codes in 4 bits (from 0000 to 1111). If we wish to represent digits as well, we need 26 + 10 = 36 codes. Now five bits are not enough, and we must take the power of 2 next higher than 36, that is 64 = 2^6. There are 64 possible codes in six bits, and six bits are what is used on many big computers – the UNIVAC 1106, the CDC 6400, and the obsolete IBM 7094. (The PDP-10, DEC's biggie, has two character code schemes; one uses six bits, the other uses seven.) Once we have 64 codes, of course, we can represent characters other than letters and digits, such as +, –, *, /, =, parentheses, period and comma, and so on – known as *special characters*. Where five bit codes are used, the special characters include *shift characters*, analogous to the shift key on a typewriter,

enabling us to pass from one group of 32 codes (including the shift characters themselves) to another such group and back.

Once we know how many bits are in a character, the choice of the actual character codes is still quite variable from one computer to another. There is a "standard" code called ASCII, or American Standard Code for Information Interchange. But this, as its name suggests, is a standard code for information *interchange* (between different computers) only, and many individual computer makers continue to use their own code schemes.

All of the codes in common use, however, share certain characteristics. One of the most important of these is that the codes for the digits are all consecutive. That is, whatever the code for zero is (and this is quite variable), the code for 7, say, is 7 more than the code for zero. This is quite helpful to us in finding the binary equivalents of integers. Another common characteristic of character codes is that the codes for letters of the alphabet are given in numerical order (although not always consecutively). Thus the code for T, for example, will be greater than the code for R, because T follows R in alphabetical order; but it will not always be true that the code for T is 2 more than the code for R.

A sequence of characters is given in a character array. On a byte machine, character arrays are easy to index. As soon as we have loaded the first character in our array into a register, we add 1 to our index register (or indirect address location) and we are immediately set up to load the next character. If we are working on a machine which holds more than one character per word – such as a 16 bit or 18 bit machine – our best course, when processing character strings (of limited size) is usually to unpack them into a word array in which one character is contained in each word. This is illustrated, for a 16 bit machine, in figure 1. After unpacking, the characters may be processed in the same way as given above.

We will thus have an index in our program which is initialized to point to the first character in our array, and which is incremented, after we are through processing that character, to point to succeeding characters in the array. Let us now turn to the question of how these characters should be processed.

Suppose, first of all, that we load a character into a register and discover that it is a *digit*. Our first job should be to find out whether any of the characters immediately following this one are also digits. Since numbers are stored internally in binary form in almost all computers, a string of digits

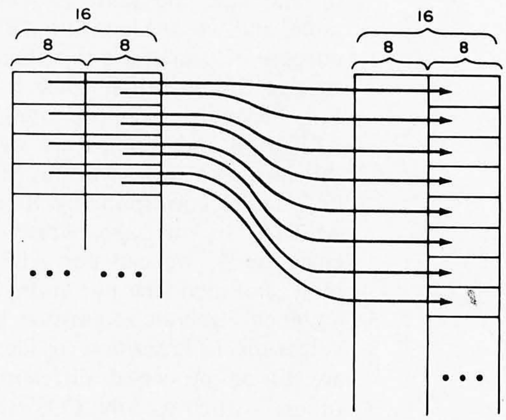


Figure 1: Unpacking characters on a 16 bit machine.

representing an integer will have to be converted to this form for further processing. Let us assume that we have a cell called NUMBER which is initialized to zero. Then our algorithm for finding the binary value of a string of digits is as follows:

1. Check the current character. If it is not a digit, stop.

2. Multiply NUMBER by 10; add the current character; and subtract the character code for zero.

3. Advance to the next character in the string and go back to step 1.

Thus for the character string 327, we perform 0 X 10 = 0, 0 + 3 = 3; then 3 X 10 = 30, 30 + 2 = 32; and finally 32 X 10 = 320, 320 + 7 = 327 – all in internal binary form. Each time we add the value of the next digit, which is equivalent to adding the character code for the digit itself and then subtracting the character code for zero.

Now suppose that, instead of a digit, we find the character code for a *letter* of the alphabet. The normal rule here is to look for letters and digits following this letter and to keep them in a string. Once we have found the first character that is neither a letter nor a digit, the letters and the digits that we have gathered so far constitute an *identifier*, which we may process further in a number of ways, depending on the way in which we are processing algebraic expressions.

As an example, consider the expression ALPHA*BETA+GAMMA*DELTA. We load the first character, namely A, into a register. Since this is a letter, we keep looking for

letters and digits, and we find L, P, H, and A. All of these characters are kept in a string. When we get to the *, the characters we have kept in our string so far constitute the identifier ALPHA. How we process this identifier depends on what it is supposed to mean. Is it a constant with a defined value? In that case the value is presumably in a table, and we can look it up. Or perhaps the purpose of processing this character string is to give this identifier some constant value. For example, suppose the string were ALPHA=3 and suppose we were processing this in such a way as to put the the value 3 in a table, corresponding to the identifier ALPHA. In this case, when we reach the character =, we can put ALPHA into our table, and then later put in the value.

Much algebraic expression processing involves *special identifiers*, or identifiers which are to be processed differently from the others – such as SIN, COS, and SQRT, or, perhaps, IF, STOP, and GOTO. All special identifiers should be collected into a table, and every time we recognize an identifier – that is, at the point in our program at which we have encountered a character that is not a letter or a digit, so that we know that the characters we have accumulated so far constitute an identifier – this table should be searched, to see whether any of its entries are equal to the current identifier. For each special identifier, we will then have a subroutine, or a section of our program, which handles it.

Let us now see what happens when, in the processing of our character string, we encounter an *operator* – a character such as +, –, *, /, or parentheses. This is where we use the Bauer-Samelson algorithm.

The Bauer-Samelson algorithm uses two *stacks* – one for operators and one for operands. Many programmers, although they understand the basic idea of a stack, have never actually written a stack-oriented program. The easiest way to do this is by using an array – call it S – together with a *current length* for the array, which we may call LS. At the start of our program, LS is set to zero. To put the quantity X on top of the stack, or, as we call it, to *push down* X on the stack, we perform

```
IF LS = MAX
        THEN GOTO OVERFLOW
LS: = LS + 1
S(LS): = X
```

where MAX is the dimension of the array S, and we transfer to OVERFLOW if we have *stack overflow*, that is, if the stack has grown too large. Pushing down X on the stack, of course, puts X on top of the stack while preserving all quantities on the stack that were previously put there. To take the quantity X off the top of the stack, or, as we say, to *pop up* X from the stack, we perform

```
IF LS = 0 THEN GOTO EMPTY
X: = S(LS)
LS: = LS – 1
```

where we transfer to EMPTY if the stack was empty when we tried to pop it. (This is usually not an error condition, but normally means that our stack process has terminated.) Of course, in the Bauer-Samelson algorithm, since we have two stacks, we may call them S1 and S2, with corresponding current lengths LS1 and LS2, both of which are initially set to zero.

In order to follow the operation of the Bauer-Samelson algorithm, we shall have to understand the basic idea of *precedence of operators*. Taking the five operators +, –, *, /, and ** (the last of these denoting exponentiation), we shall assign to each one a number called its *precedence*, as follows:

+	1
–	1
*	2
/	2
**	3

The easiest way to understand precedence is to consider what would happen if we didn't have it. For example, let us look at the expression 2*5+3*4. Suppose we tried to find the value of this expression in the following way: 2 times 5 is 10, plus 3 is 13, times 4 is 52. Clearly, this would be the wrong answer. What we want to do is to perform the multiplications first, namely 2 times 5 is 10 and 3 times 4 is 12, and then add together the resulting values, 10 and 12. Since we perform the multiplications *before* the addition, we say that multiplication (as an operator) has *higher precedence* than addition. The numbers which we have assigned to the operators reflect this fact; that is, 2 (the precedence of multiplication) is higher than 1 (the precedence of addition).

We shall now describe the basic operations of the Bauer-Samelson algorithm. The algorithm proceeds by scanning over the characters of the given string from left to right. Every time we encounter an *operand* – a constant or a variable – while we are doing this, we place it on the operand stack S1. Every time we encounter an *operator*, we try to place it on the operator stack S2. This is not done unless the *precedence test* is satisfied; that is, unless the precedence of the given operator is higher than that of the

operator at the top of the stack (or unless the operator stack is empty). If the precedence test is not satisfied, we remove an operator from the top of the stack S2, remove its operand or operands from the top of S1, calculate a result, and put this result back on S1. This is called *unstacking* an operator. We continue to unstack all operators from S2 until the precedence test is satisfied. When we reach the end of the entire original string, we unstack all operators from S2.

An example should make this clear. Suppose we have the string 2*5+3*4 mentioned earlier, and we are trying to calculate its value, which is in this case not 52 but 22. In the following analysis, we shall denote the contents of a stack by several quantities in parentheses, with the *last* of these quantities denoting the top of the stack. Thus (10,3,4) as the contents of the stack S1 means that the number 4 is at the top of this stack. The Bauer-Samelson algorithm proceeds in this case as follows:

1. The 2 goes on the stack S1.
2. The first * goes on the stack S2. (The precedence test is satisfied, since the stack S2 was empty.)
3. The 5 goes on the stack S1, which now contains (2,5).
4. We cannot put the + on the stack S2, because the precedence of + is lower than that of *. Therefore we unstack the * from the operator stack S2. This means: we take * off the stack S2, leaving S2 empty; we take its operands off the *top* of S1 (that is, we take 5 and 2 off S1); we calculate the result, namely 2*5 = 10 (the *second* operand of any operator is always removed from the stack first); and we put the 10 back on the operand stack S1, which now contains 10 and nothing else.
5. We are supposed to keep unstacking operators until the precedence test is satisfied. At this point, however, the precedence test is in fact satisfied, since the operator stack is empty, and we may therefore place a + on the operator stack and continue.
6. The 3 goes on the stack S1, which now contains (10,3).
7. The * goes on the stack S2, which now contains (+, *). The precedence test succeeded in this case, since the precedence of * is higher than that of +.
8. The 4 goes on the stack S1, which now contains (10,3,4). We are now at the end of the original string, and it is time to unstack all the operators from the stack S2.
9. The operator at the top of S2, namely *, is taken off this stack. Two operands are taken off the top of S1, namely 4 and 3; this leaves 10 on S1. The result, namely 3*4 = 12, is calculated and placed back on S1, so that S1 now contains (10,12).
10. The operator at the top of S2, namely +, is taken off this stack. Two operands are taken off the top of S1, namely 12 and 10; this leaves S1 empty. The result, namely 10+12 = 22, is calculated and placed back on S1.
11. The operator stack S2 is now empty; the Bauer-Samelson algorithm has finished; and the answer, namely 22, is on the operand stack S1. (Unless there has been an error, the Bauer-Samelson algorithm will always end with exactly one quantity on the operand stack, and this quantity will be the final result.)

This is the basic Bauer-Samelson algorithm. It may now be modified and extended in a number of ways.

Let us first consider parentheses. A *left* parenthesis is treated as an operator. It is always placed directly on the operator stack without making the precedence test; that is, it is treated as if it had the highest precedence. Once it is on the operator stack, however, it is treated as if it had the *lowest* precedence; that is, *any* other operator is placed directly above it on the stack, or, to put it another way, the precedence test always succeeds if there is a left parenthesis at the top of the operator stack.

A *right* parenthesis is treated somewhat like the end of the expression. We unstack all operators on the operator stack until we come to a left parenthesis, which we remove from the operator stack and continue to scan the given string. If there is no left parenthesis on the operator stack, there were too many right parentheses in the original expression. Conversely, if we come to the end of our string and start unstacking operators, and one of these is a left parenthesis, then there were too many left parentheses in the original expression.

As an example of the use of parentheses, we consider the expression 2+4*(5–(6–3))/8, the value of which is 3. We shall again "walk through" the Bauer-Samelson algorithm as it scans this string. This time, however, we shall use an abbreviated notation. In the second column below, marked action, we use one of the following codes:

O (Operand) – An operand is placed on the operand stack.

S (Succeed) – The precedence test succeeds, and therefore an operator is placed on the operator stack.

U (Unstack) – The precedence test fails (or else we are at the end of the expression, or at a right parenthesis), and thus an operator is unstacked.

Table 1: Calculation of the value of 2+4(5–(6–3))/8.*

Current Character	Action	Operand Stack	Operator Stack
2	O	(2)	Empty
+	S	(2)	(+)
4	O	(2,4)	(+)
*	S	(2,4)	(+,*)
(	L	(2,4)	(+,*,L)
5	O	(2,4,5)	(+,*,L)
–	S	(2,4,5)	(+,*,L,–)
(	L	(2,4,5)	(+,*,L,–,L)
6	O	(2,4,5,6)	(+,*,L,–,L)
–	S	(2,4,5,6)	(+,*,L,–,L,–)
3	O	(2,4,5,6,3)	(+,*,L,–,L,–)
)	U	(2,4,5,3)	(+,*,L,–,L)
	R	(2,4,5,3)	(+,*,L,–)
)	U	(2,4,2)	(+,*,L)
	R	(2,4,2)	(+,*)
/	U	(2,8)	(+)
	S	(2,8)	(+,/)
8	O	(2,8,8)	(+,/)
End	U	(2,1)	(+)
	U	(3)	Empty

L (Left parenthesis) – A left parenthesis is placed on the operator stack. (This is denoted by L in table 1.

R (Remove left parenthesis) – A left parenthesis is removed from the operator stack (this happens after unstacking, when the current character is a right parenthesis).

The operation of the Bauer-Samelson algorithm in this case can now be expressed by means of table 1.

Of course, the "current character" column in table 1 takes advantage of the fact that every operator and every operand in our example program consists of a single character. In a more general case, this column would be headed "current operator or operand."

Let us now consider unary operators. Superficially, there is no difference between a unary and a binary operator from our point of view, except that when we unstack a unary operator we must remove only *one* operand, rather than two, from the operand stack. However, when we put the unary minus sign on the operand stack, we must be careful to identify it as a unary, rather than a binary minus sign, so that we know how many operands to take off the stack S1. This in turn means that we are going to have to be able to tell the difference between a unary and a binary minus sign as we are scanning our string.

The simplest way to do this is to think of our Bauer-Samelson algorithm as having two basic labels, which we shall call L1 and L2. We start off at L1 (after all necessary initializations). When we are at L1, we are expecting to find an *operand*. If we find one, we put it on the stack S1, and go to L2. When we are at L2, we are expecting to find an *operator*. If we find one, it will be a binary operator; we put it on the stack S2 (after any necessary unstacking) and go back to L1. But if we are at L1 and we find an operator, it must be a unary operator; we put it on S2, after unstacking if necessary, and then return to L1.

Suppose now that we find a right parenthesis. Then we must be at L2 (if we are at L1, we have an error in the string we are scanning). We perform all necessary unstacking, remove a left parenthesis from the operator stack as described above – and then return to L2, since we are now expecting a binary operator. If we find an operand at L2, this is also an error condition. If we find a left parenthesis (of the type that we have so far described), we should be at L1; we put it on the operator stack and then return to L1, since we are again expecting to find an operand.

This interplay between L1 and L2 may be illustrated by the following example, containing two unary and two binary minus signs, in addition to parentheses. For the moment, we shall consider a unary minus sign to have precedence equal to 2. A binary minus on the stack S2 will be denoted by '–', a unary minus by U, and a left parenthesis by L. The codes in the "action" column are as in the preceding example. The string to be scanned is –5–(–3–4); its value, which is 2, is calculated by the Bauer-Samelson algorithm as in table 2.

Table 2: Calculation of the value of –5–(–3–4).

Current Label	Current Character	Action	Operand Stack	Operator Stack
L1	–	S	Empty	(U)
L1	5	O	(5)	(U)
L2	–	U	(–5)	Empty
		S	(–5)	(–)
L1	(	L	(–5)	(–,L)
L1	–	S	(–5)	(–,L,U)
L1	3	O	(–5,3)	(–,L,U)
L2	–	U	(–5,–3)	(–,L)
		S	(–5,–3)	(–,L,–)
L1	4	O	(–5,–3,4)	(–,L,–)
L2	)	U	(–5,–7)	(–,L)
		R	(–5,–7)	(–)
L2	End	U	(2)	Empty

It is, incidentally, a matter of controversy as to what the precedence of the unary minus should be. It should clearly be lower than that of exponentiation (thus –X**N is clearly –(X**N), and not (–X)**N) and higher than that of addition (thus –X+Y is clearly (–X)+Y, and not –(X+Y)). What about –X*Y, however? The two expressions (–X)*Y and –(X*Y) are equal, and the same is true of (–X)/Y and –(X/Y). It is not clear which choice leads to the greatest efficiency of calculation.■

BYTE'S BITS

New England Seminar on Computers in the Laboratory

The New England Section of the Society for Applied Spectroscopy and the Analytical Section, American Chemical Society (New England) announce a seminar series, Automation and the Laboratory - 1976. This seminar, open to non-members of the sponsoring organizations, is aimed at the scientist, professional researcher or engineer with little or no formal training in computers and digital electronics. The course is designed to cover laboratory automation topics ranging in scope from the smallest microcomputer systems to traditional large scale installations. Speakers are from the local universities, industry and computer publications.

All sessions will be Thursday nights, 6:30 to 9:30, at the Hartwell Avenue facility of IL Inc., Lexington MA 02173. The fee for individuals who are not members of the sponsoring societies is $30 for the whole series, $5 for individual lectures. Each session will be divided into two segments separated by a coffee break with refreshments. For details contact Marti Bancfroft at:

IL Inc.
Jonspin Road
Wilmington MA 01887

or, by phone during business hours at
617-658-5125 Extension 40

If you have been out of school five years and are a little frustrated by computereeze, this seminar is for you. It should be especially appropriate if you are considering the use of laboratory data processing equipment.

MICROCOMPUTER PROFILE
(A One Day Technology Update)

A survey and update on microcomputers and how they make it easier to use microprocessors, presented with hands-on demonstrations and important survey materials. Seven offerings during March 1976: Los Angeles, Chicago, Ottawa, Boston, Washington DC, Dallas, and San Diego. Fee: $165 includes survey papers, course notes, reprints, coffee breaks and luncheon. For information: Technology Service Corporation, 2811 Wilshire Boulevard, Santa Monica CA 90403. Telephone: (213) 829-7411.

An Update on the SCCS Computer Purchase

Hal Lashlee, SCCS treasurer, reports the following progress on the Los Angeles group's DEC LSI-11 purchase. First, the SCCS has changed its commission to 7% of the purchase price (larger donations are solicited). Second, the first order of 50 machines has been placed with DEC now that the minimum number has been reached. Third, SCCS now has an OEM arrangement with DEC and will continue to coordinate purchases of the LSI-11 and related equipment. Individuals interested in an LSI-11 should contact Hal at the SCCS post office box address:

Southern California Computer Society
PO Box 987
South Pasadena CA 91030

Data Paths

Gary Liming
3152 Santiago Dr
Florissant MO 63033

Data transmission usually brings to mind terminals, telephone lines, satellites, and large computer centers. Computer links in retail stores, banks, airlines and government agencies are becoming more and more widespread. Such large scale operations can easily cost millions of dollars and are thus out of the range of the hobbyist. The prospect of linking home systems across distances for program swapping and interactive games will undoubtedly become more a possibility as the technology improves.

However, data transmission in a broader sense doesn't have to mean large networks of computers and remote terminals. It is defined as the process of sending error free bits from one place to another, and applies to all digital systems regardless of complexity. In this article we illustrate some data transmission principles applied to hobby system design, over distances ranging from the length of printed circuit foil runs on a circuit card to the extremely long distances involved in phone or radio links.

Data transmission in a broader sense doesn't have to mean large networks of computers and remote terminals.

Communication Theory

Data transmission is part of the broader subject of communication theory which is used to analyze communication systems. Any communication system has three parts: a message *source*, a *medium*, and a *receiver*. To communicate, information of some kind must be transferred. Information is defined simply as an orderly representative signal. Orderly means that the signal is sent in a known format which can be interpreted and decoded by the receiver. Representative means that there is agreement between source and receiver upon what the signal will mean. A signal could be a series of printed characters, a bell, a whistle or even a color. The smallest unit of information is the bit, representing only an on off or yes no condition. One or a series of these fundamental bit signals makes up the message in digital communications.

Any medium that can transfer a message has limits, and the medium within these limits is called the *channel*. The limits which define a channel might be physical properties such as the technologically available bandwidth, or human defined limits such as an arbitrary FCC ruling that a radio station is allocated a particular set of frequencies with a prescribed bandwidth for its signals. *Noise* is defined as any signal that interferes with the message, like radio static or dirt on a camera lens.

A communication that works one-way, or does not allow information to be mutually exchanged is called a *simplex* transmission or communication. If information can be exchanged, it is called a *duplex* system. There are two kinds of duplex systems: If information can be sent between two points simultaneously, it is called a *full duplex* system; if the information can be transferred in both directions but not at the same time, it is called – you guessed it – *half duplex*. Figure 1 illustrates the various kinds of communications exchanges.

Let's apply this to a simple example – consider the page you are looking at. The author is the message source, you are the receiver, and paper and ink are the medium. The size of the page sets the channel limits, and ink blots or printing errors comprise noise. Communication is simplex. When a reader replies, it has become half duplex.

This point of view can be applied at different levels to your system design. Integrated circuits, printed circuit boards, peripherals and terminals can all be considered sources and receivers. They all use the bit as the common unit of information.

An important factor in data communications is the *data or transmission rate* at which the bits are transferred. This is measured, naturally, in bits per second (abbreviated b/s). It is on this simple point that many newcomers first get into trouble by using the term baud. Baud has a different meaning which can be ambiguous, as we will see when we look at modulation methods and modems.

Another important parameter of information transmission is the *error rate*, measured by the number of bits in error out of the

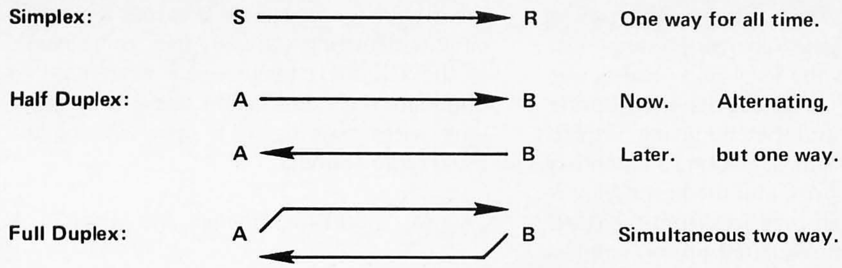


Figure 1: The terminology describing modes of communication between sender and receiver depend upon who is sending data and when the data is sent.

Data transmission is the process of sending error free bits from one place to another.

total number transmitted. If your computer processes instructions at 500 kilobytes per second which equals 4 megabits per second, an error rate of one in 10 million (10**—7) will give you an error on the average of every 2.5 seconds. Clearly, what is a tolerable error rate depends on the transmission rate. A central processor which has errors every 2.5 seconds is not very usable.

Medium Characteristics

To transfer data, the hobbyist can use any medium that will support a bit stream. There isn't any reason why you couldn't take a serial interface and hook it up to modulate a laser beam. However, since most of the transmission done by the hobbyist uses conductors, let's first look at small gauge wire used as a communication channel.

As the data rate through the wire increases, the bit stream begins to look like an AC signal passing through a transmission line, and must be treated as such. This *is not* necessarily related to a reversal of current flow, like AC, but is due to the fast rise and fall times of the pulses.

Therefore, the channel must have a fairly wide band of frequencies it can pass (*bandwidth*). It must pass an AC signal with approximately the same rise and fall times as the pulse and the flat portion of the pulse, which is essentially AC at zero frequency.

Conductor properties such as resitance, capacitance, and inductance degrade the quality of the pulses. The voltage drop due to the resistance of the wire lowers the voltage of the received pulse. Capacitance between the signal wire and the ground wire shunts some of the voltage, and inductance and capacitance both provide impedance to the flow of the pulses. Noise induced from the environment and the power supply will further degrade signal quality.

Another problem amateur radio operators will be familiar with is the skin effect, where high frequency current tends to concentrate in the outer layers of the wire, increasing the effective resistance. Also, the propagation time of the pulses should be taken into account. Even though the pulses travel at near the speed of light, for 22 AWG (0.79 mmϕ) wire, the delay is about 1.5 ns/ft (4.9 ps/cm); a 100 foot coaxial cable introduces a transmission delay which is nearly a whole machine cycle delay in some high speed systems. Indeed, such transmission lines are often used as delay elements in oscilloscopes.

All these phenomena depend on the length of the wire and the frequencies of the signal. They can combine to ruin the shape of the pulse to the extent that the logic gates can become confused as to whether they are seeing a zero or a one. We conclude then that the longer the wire and the higher the transmitted frequencies, the harder it is to get an acceptable error rate.

Microtransmissions

To communicate, information of some kind must be transferred.

Armed with these characteristics and definitions, let's look at how conductors affect data transmission in a typical processor. To date hobby systems have been predominantly designed with 7400 series TTL, which can handle clock frequencies up to around 35 MHz, but are commonly clocked at around 1 MHz. At 1 MHz, characteristics like resistance and capacitance of wires are not significant for short transmissions such as chip to chip or board to board transfers. The big problem *inside* systems is induced high frequency noise due to changes of logic states. The typical TTL transition time of ten nanoseconds has a significant harmonic content well into the VHF range of 50 to 200 MHz. (This is the reason your computer can generate some powerful television interference if it is not properly shielded.) The current surges at the power connection of a TTL gate which is changing state induces a noise signal, since the power bus is typically a poor conductor of VHF.

Thus one common source of noise is a poorly designed power supply and distribution system. Because of its high speed characteristics, TTL logic is very sensitive to

What is a tolerable error rate depends upon the number of bits per second transferred.

changes in its supply voltage. The power surges of one gate changing state can momentarily drop the level of a local power distribution wire, affecting its neighboring integrated circuit and thereby giving birth to a *glitch* in the system. Detecting a glitch is a real hassle for the pros, and the best policy is to use sound design practices from the start. The design of well regulated power supplies is a significant subject in its own right, and will not be covered in this article. Home brew computer experimenters can often find excellent high current logic power supplies in surplus stores.

Noise spikes in the power wiring can also occur between chips on the power paths and can spread to other chips and boards. These noise spikes in the power wiring are induced due to the inductance of the printed circuit foils or wire wrap wires as the gates change state and draw a lot of current. Using wide flat power supply runs in the PC artwork will lower the high frequency impedance of the conductors. Problems can be further minimized by placing many small ceramic decoupling capacitors of approximately 0.01 uF between the positive power supply bus and ground. Use one decoupling capacitor for every five to ten TTL integrated circuits. Using an integrated circuit voltage regulator on each board will also help provide isolation of power supply noise between boards.

A well grounded case will greatly help reduce environmental noise. The case will also shield you from your neighbor's complaints about interference with his television reception. Another benefit of a well grounded system case or chassis is protection from static electricity. In a dry house in winter, shuffling across the room to turn on the system can wipe out some MOS chips, as I know from bitter experience.

These may not sound like important data transmission problems, but they are direct results of the same high frequency transmission characteristics which affect long wire links. Troubles that start with an improperly designed power distribution and layout scheme are hard to spot and correct, but will certainly show up in transmissions over long wires.

Macrotransmissions

Macrotransmission problems occur between central processors and peripherals. The transmission line characteristics become important: If the length of the wire approaches the order of magnitude of the wavelength of the signal, transmission line effects are a potential source of problems. This phenomenon occurs in short wires at high frequencies, and in longer wires at lower frequencies. As mentioned previously, the frequency characteristics of TTL logic circuits changing state – VHF components in the 100 MHz range – are what tend to dominate the transmission line properties of long wires carrying TTL signals. Using the usual radio formula,

$$\lambda = 300/f \ (\lambda \text{ in meters, f in MHz})$$

gives wavelengths for the high frequency components of a TTL state transition which are in the vicinity of three meters at 100 MHz. Thus cables with lengths of one or two meters should exhibit many of the properties of transmission lines when they carry standard TTL signals. Note that this property primarily depends upon the transition time, and is independent of the actual number of transitions per second. By slowing the transition time by a factor of 100 to one microsecond or more, transmission line effects will not begin to occur until cables of 100 meters or more are considered. Given some arbitrary length of cable, the alternatives open are to take into account transmission line behavior through impedance matching techniques, or to slow down the signals so that transmission line effects are no longer a consideration. Since the latter option produces a non-TTL signal because it changes state too slowly, let's turn attention to methods of compensating for transmission line behavior.

As a simple example, consider two parallel wires. Each wire has the properties mentioned before, and is represented in figure 2. In order for the pulses to travel through the conductor with minimal losses in signal quality, each end of the cable must be terminated properly. Termination of the

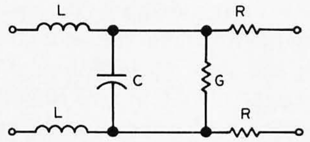


Figure 2: Symbolic representation of parallel wire transmission. The system is symmetric, so it does not matter whether the left terminals are at the source and the right terminals are at the receiver or vice versa. The symbols used in the diagram are as follows:

L, inductance of the wire.
R, resistance of the wire.
C, capacitance between the wires.
G, high resistance leakage path between the wires.

ALTAIR CONVENTION

Dear Friends,

You are invited to attend the first annual WORLD ALTAIR COMPUTER CONVENTION. This exciting event will be held at the new MITS factory on Saturday and Sunday, March 27-28, in Albuquerque, New Mexico.

Hobby clubs and individual Altair owners have been asked to bring their Altairs to Albuquerque to set up demonstrations. Thousands of dollars worth of Altair equipment will be awarded to the users with the most innovative demonstrations.

Seminars will be conducted by MITS engineers, MITS software writers, and by some of the leading figures in the home computer field.

Since the new MITS factory is located close to the Albuquerque Airport, it will be very convenient for out-of-town guests to attend the convention. The entire Albuquerque Airport Marina Hotel has been reserved for this occasion. This hotel is across the street from the terminal building and just a short walk away from MITS.

Attendence to the convention will be free (not counting travel and hotel costs) and MITS will provide door prizes and at least one free luncheon.

Call or write for hotel and convention reservations. Or for more information:

WACC/MITS
2450 Alamo SE
Albuquerque, NM 87106
phone 505-265-77553 or 262-1951

With the help of the thousands of Altair Users and friends of Altair, it is hoped that this convention will be one of the most exciting computer events in the industry.

Thanks from

MITS
"Creative Electronics"

*Altair is a registered trademark of MITS, Inc.

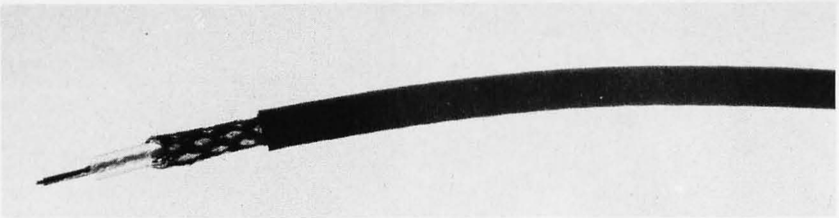


Photo 1: Coaxial cable consists of a central conductor, an outer conductive braid, and a protective coating. It is bulky and expensive, but it has good characteristics as a transmission line for data.

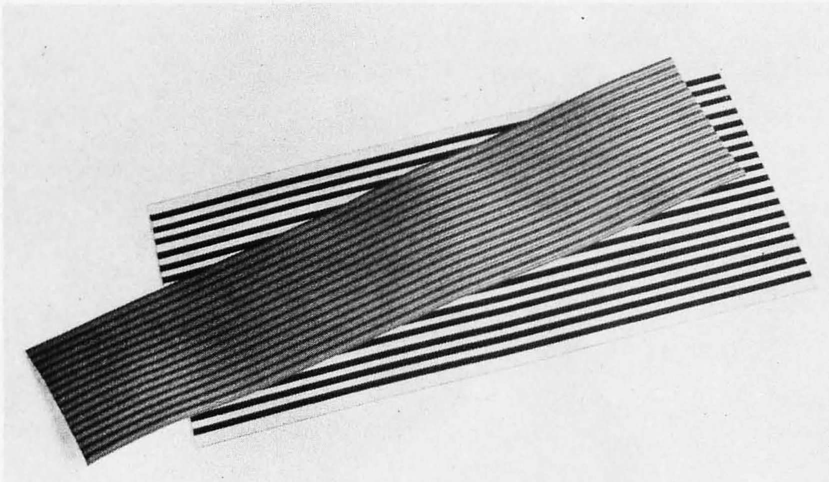


Photo 2: Two examples of ribbon cable. The lower example is a surplus item consisting of flat copper conductors (similar to PC lamination) embedded in a plastic carrier. The upper example is a more conventional cable intended for assembling to a special dual in line package (DIP) plug.

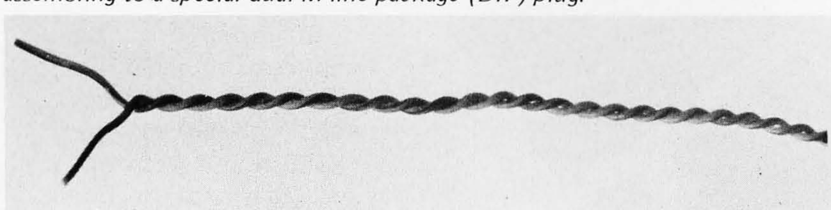


Photo 3: Twisted pair cable is the only good data transmission line which can be easily fabricated at home. Here is an example made using an electric drill to do the twisting.

line involves matching the *characteristic impedance* of the wire with the *impedance* of receiver and transmitter.

As a pulse is sent to the other end, the energy of the pulse is dissipated by the termination of the wire. If the wire is not terminated properly, a *reflection* of the pulse will travel back to the source, and a condition called *ringing* will occur.

It is for this reason that flip flops should never be used to directly drive a line of significant distance. Ringing or noise spikes could occur on the line and enter the flip flop circuit and change its state.

Typical 7400 gates have an impedance of 100 O in the high state and nearly 0 O in the low state. Almost all newer small scale TTL integrated circuits are diode clamped, preventing most ringing on the inputs. This allows wires to go about 5 feet between gates without using external impedance matching techniques, and assumes a relatively high speed and constant impedance line. If a standard TTL gate is used as the transmitter in a data link, fan out rules must be observed to supply sufficient current. To raise the output voltage of the pulse, a 2.2 kO resistor can be connected between the output and the 5 V source. This pullup resistor raises the output pulse to a full 5 V and reduces the chances of noise affecting the line.

For longer runs at high speeds, a TTL line driver chip like the 74128 can be used to provide more current to the line. For even longer runs or in critical applications special chips like the Signetics 8T13 and 8T16 are used as drivers and receivers to insure a low error rate. The maximum length for these transmissions depends on the type of wire being used.

Coaxial cable is one of the best cables to use for long distance transmission of digital data. It has a center conductor set in a non conductor with a metallic braid or foil (the shield) wrapped around it. The shield is used as the ground return and for protection from external noise. Photo 1 illustrates a typical coaxial cable, cut so its construction can be seen. Cables with a nominal characteristic impedance of around 100 O are normally used in order to match gate terminations. Coaxial cable has the disadvantage of being inflexible and bulky, especially if many wires have to be terminated in a small area. An even worse disadvantage is its high cost. Coax is usually used when other wires aren't suitable.

Flat ribbon cable, which usually has every other conductor grounded, provides a constant impedance and reduces the chance of wires inducing signals on each other. Ribbon cable for the hobbyist is still rather expensive, and special connectors generally must be used. Photo 2 shows two types of ribbon cable: flexible copper strips in plastic, and multiple stranded wires.

Twisted pair wire is the most cost effective transmission line for long runs in hobby systems. This kind of cable can be fabricated at home using an electric drill. In multipair cable, each pair should be used as a single signal path, with one wire grounded. The rise time characteristics of the pair are determined by the conductor size and tightness of the twist. For a 100 O cable, the wire should be 22 to 24 AWG (stranded) with about three turns to the inch. Multipair wire is available at many surplus houses, and is generally a bargain. Photo 3 illustrates a typical home made twisted pair.

Good old hookup wire is the most susceptible to noise and usually has a highly

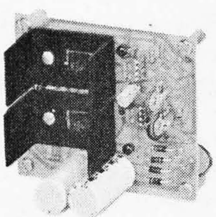
COMPUTER EXPERIMENTER SUPPLIES

FACTORY FRESH—PRIME QUALITY
PERFORMANCE GUARANTEED

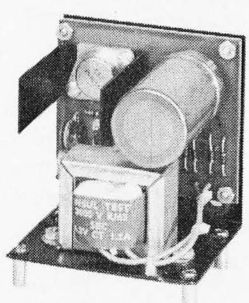
MICROPROCESSORS AND MEMORY

8008	$ 35.00	Commercial Grade—up to 35° C.
8080	135.00	These units are factory fresh, full spec devices.
2102	3.50	
2102-2	4.50	

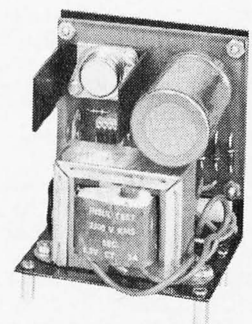
COMPUTER GRADE REGULATED POWER SUPPLIES
All units are short-circuit proof, fold back current limited and with over-voltage crowbar protection.



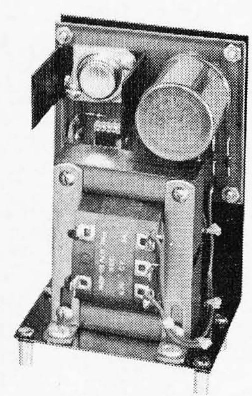
MD-15
±15 Volt at 200MA
Dual Tracking
$30.00



MD-5-1
+5 Volt at 1 Amp
$24.50



MD-5-3
+5 Volt at 3 Amp
$34.50



MD-5-6
+5 Volt at 6 Amp
$44.50

MICRO COMPUTER SUPPLY COMBINATIONS

For the 8008
MD-08—+5 volt at 6 amp, -12, -9 at 200 ma $75.00
For the 8080
MD-80—+5 volt at 6 amp, ±12v at 200 ma ... $75.00
For the Fairchild F-8
MD-8—+5 volt at 6 amp, +12 v at 200 ma ... $65.00
For the M6800
MD-5—+5 volt at 6 amp $44.50

All units are short circuit proof, fold-back current limited and with over voltage crowbar protection.

TTL INTEGRATED CIRCUITS

All devices are factory fresh, full spec units.

7400	.23
7404	.25
7442	.60
7447	.95
7448	.95
7475	.60
7490	.60
7493	.60
74125	.55
74126	.55
74192	1.10
74193	1.10

GUARANTEE
Most devices shipped within 24 hours. If not shippable within 2 weeks payment refunded. Performance guaranteed on all units for 30 days. Defective parts replaced at no charge. NOTICE: This warranty applies **only** to parts that have not been soldered. You must use sockets for your incoming inspection tests.

All Prices Subject to Change Without Notice
Minimum Order $10.00
Add $1.00 to Cover Postage and Handling
Send Check or Money Order (No C.O.D.) To:
N. J. Residents Add 5% Sales Tax

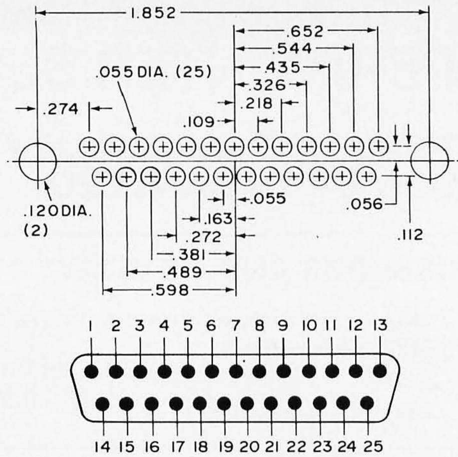
MICRO DIGITAL CORP.

BOX 413, EDISON, NJ 08817 • (201) 549-2699

unpredictable characteristic impedance. Hookup wire should be used only for short runs between boards and as on board jumpers.

A well grounded case . . . will help shield you from your neighbor's complaints about interference with his television reception.

When wires get long enough to look like transmission lines, termination and impedance matching become important.



RS-232 PIN ASSIGNMENTS

Pin	Name	Function
1	FG	Frame Ground (not switched)
2	TD	Transmit Data
3	RD	Receive Data
4	TRS	Request To Send
5	CTS	Clear To Send
6	DSR	Data Set Ready
7	SG	Signal Ground
8	DCD	Data Carrier Detect
9		Positive DC Test Voltage
10		Negative DC Test Voltage
11		Unassigned
12	(S)DCD	Secondary Data Carrier Detect
13	(S)CTS	Secondary Clear To Send
14	(S)TD	Secondary Transmit Data
15	TC	Transmit Clock
16	(S)RD	Secondary Receive Data
17	RC	Receive Clock
18		Receiver Dibit Clock
19	(S)RTS	Secondary Request To Send
20	DTR	Data Terminal Ready
21	SQ	Signal Quality Detect
22	RI	Ring Indicator
23		Data Rate Select
24	ETC	External Transmit Clock
25		Busy

Figure 3: The commonly used RS-232 electrical interconnection for data transmission is shown here in the form of pin assignments for the typical D connector. A typical part number for the connector is DB-25P (plug) and DB-25S (socket) made by Cinch.

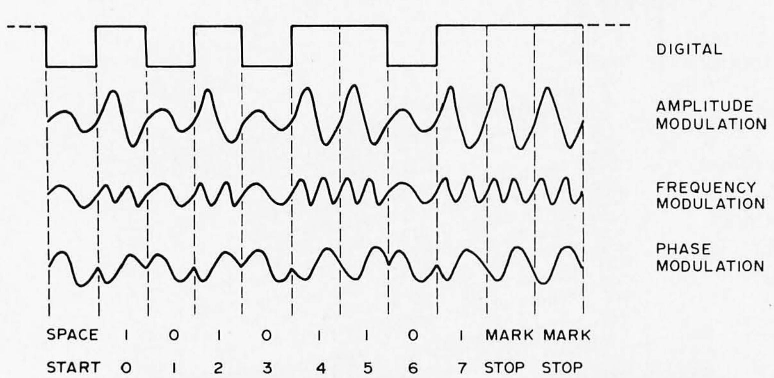


Figure 4: Two level asynchronous modulation, shown for the ASCII character "5" along with typical modulated waveforms for different methods of modulation.

The Long Haul

At distances greater than your next door neighbor's house, it begins to become impractical to use your own wire between systems. This is primarily due to a very important conductor property – cost. Copper is not cheap, and stringing wire all over the place will give hobbyists a bad name. However, the phone company has already done this and will provide service to you if you pay the price.

Another alternative is using a different medium, like radio waves. It will be interesting to see how many people suddenly want FCC ham licenses in order to play interactive TV games and exchange software.

Both of these methods have a major drawback; they will not directly pass a digital signal. The telephone system was designed to carry audio frequencies. Phones just cannot carry DC level signals. Radio frequencies are at the other end of the spectrum and certainly won't support a usable DC coupled logic signal.

However, the pulses of digital transmission can be superimposed on an AC signal that is within the bandwidth of the channel being used. The process of superimposing one signal on another is called *modulation*. A device that will translate the digital bit stream to an encoded analog signal for transmission and analog to digital for reception is called a *mod*ulator-*dem*odulator, or *modem*.

Since it is impractical to have eight telephone lines or eight radio frequencies transmit in parallel, a conversion to serial must be done. This can be accomplished by writing the conversion in software, but is more cost effective by using an integrated circuit called a UART (Universal Asynchronous Receiver-Transmitter). This chip takes the data in parallel form and converts it to serial at a rate specified by external components. For a more detailed discussion of the serial interface and UARTs, see "The Serial Interface" by Don Lancaster in the September 1975 issue of BYTE.

There is a standard for interfacing serial data transmission between peripherals, systems, and modems. The Electronic Industries Association (EIA) of America has, by consent of various manufacturers and users, standardized a 25 pin connection with appropriate signal levels called the RS-232 interface. There is also a new EIA standard that has been introduced, called the RS-422 standard that is more suitable to TTL.

However, since almost all modems and peripherals now available and all surplus items are likely to use the RS-232 standard, it will remain the most significant to hobbyists for some time.

A commonly used RS-232 connector along with pin assignments is shown in figure 3. The signals that appear on the signal pins must be bipolar with 3 V to 25 V representing a logical one and −3 V to −25 V being a logical zero. There are chips available to do this conversion from TTL, the 1488 and 1489. If you are going to use commercial RS-232 equipment, you should expect to provide this interface; and you'll also need the positive and negative supply voltages. Keep in mind that in common practice many of the pins in figure 3 are not used. The most important lines on the RS-232 interface plug are the grounds, the transmit data (TD) and receive data (RD). In many instances it is sufficient to use only these lines, especially if you are just experimenting with an RS-232 peripheral. Note, however, that some terminals require inputs for one or more additional pins, many of which can simply be wired to the RS-232 logic zero or logic one lines (−12 V or +12 V, for example).

Data Modulation

There are three basic techniques for modulating an analog signal and many different variations of these. An analog signal that is to be modulated by the data is called a *carrier*, and the carrier has three basic characteristics that can be varied. If the amplitude, frequency, or phase is varied in step with the bit stream, modulation occurs. An example of each technique is shown in figure 4.

Amplitude modulation is seldom used in modems because of its high susceptibility to noise and attenuation, but the technique is used in some magnetic tape encoding schemes. Frequency modulation is a more common technique. The example shown in figure 4 uses one frequency to represent a 0 bit and a higher frequency to represent a 1 bit. This particular method is called *frequency shift keying* (FSK).

The phase modulation example in figure 4 shows a two level coding scheme with each 180° phase shift triggering a logical state change.

The number of times the signal is varied each second is called *Baud* or *Baud rate*. Suppose you were designing a modem using phase modulation. You could, for example, divide the possible phase shifts into 45° each, having eight possible phase shifts for each signal change. A 45° shift would represent a group of three bits, namely '000'. 90° would represent '001', 135° = '010', 180° = '011', etc., up to 360° for '111'. In this case a signal changing 100 times a second, or at 100 Baud, would actually transfer data at 300 b/s. This method of one signal change representing more than one bit is called *multilevel encoding*. It is in principle the way some commercial high speed modems function.

In order to obtain a fair amount of accuracy in transmitting data from a transmitter to a receiver, it is necessary to keep the two systems in step with each other. There are two common methods to do this, called *synchronous* and *asynchronous* transmission.

Asynchronous transmission is also called start-stop transmission because each character is sent as it is created at the transmission interface. To synchronize the receiver, each character carries its own timing in the form of additional bits called start and stop bits. These give the receiver the ability to decode each bit reliably. The format of a single character is shown in figure 4 along with data for the digit 5 encoded in ASCII.

Synchronous transmission is usually associated with blocks of data, where groups of characters are sent together. A fixed speed of transmission is set by clocks or oscillators, and data bits are transferred at this rate. To provide character synchronization, usually two special synchronization characters precede the actual block of data. An end of block character follows the data to signal the receiver that all of the data in that block has been sent. An additional error detecting character may also be sent as shown in figure 5.

A baud is not necessarily a bit per second — although it might be in special cases. A baud is a signal change per second.

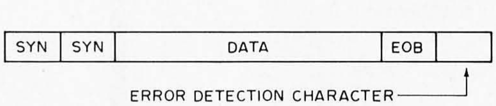


Figure 5: Synchronous transmission is block oriented and assumes highly accurate common clocking of both the sender and the receiver.

Finally, there is one other complication in using radio or telephones for your communication channel — the FCC. The FCC regulates the telephone industry by tariffs that specify the costs and types of devices that can be used with Ma Bell. They also regulate radio frequency allocation and power output and information codes that can be transmitted by radio.

There are good reasons for having these regulations followed, but they do tend to make life difficult for the hobbyist. A dedicated 100 mile phone line with a Bell modem will exceed most other system costs

for a home computer in a very short time. A ham license and suitable radio equipment are not cheap, either. Long distance real time data transmission is presently out of reach to many hobbyists.■

GLOSSARY

Asynchronous transmission: Transmission where data is sent a character at a time with synchronizing bits added. See Synchronous Transmission.

b/s: Bits per second.

Bandwidth: The width of the frequency spectrum that a channel can pass, measured in Hz.

Baud: The number of signal changes per second.

Carrier: The analog signal that is modulated by the information that it is to carry. Also, the provider of a communications channel, i.e., Ma Bell.

Duplex: Communication system that allows information to be exchanged. See simplex.

FSK: Frequency shift keying, a type of frequency modulation for digital data.

Full duplex: Communication system that allows simultaneous information exchange. See half duplex.

Half duplex: Communication system that allows information to be exchanged, but not simultaneously. See full duplex.

Modem: Modulator-demodulator. A device used to convert digital to analog signals and vice versa.

Modulation: The process of superimposing information on a carrier. See **carrier**.

Multilevel Encoding: The process of using a signal change to represent more than one bit of information.

Noise: Unwanted signals that interfere with the message.

Ringing: A condition in transmission lines where "reflections" of pulses oscillate due to an impedance mismatch.

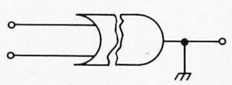
Simplex: Communication system that does not allow information to be exchanged. See duplex.

Synchronous Transmission: Transmission where the bit rate is clocked. Usually associated with block transmission. See asynchronous transmission.

Chips Found Floating Down Silicon Slough

Roy H Trumbull
833 Balra Dr
El Cerrito CA 94530

The state of the Art is changing rapidly. In fact I ran into him in New Mexico last month. Seems he had just gotten back from China where he had seen their latest computer. It was really fantastic, but they still have a problem with noise from the beads. I asked Art to clue me in on the latest devices coming out of research and these are the ones he told me about:

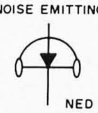


DON'T GATE

The Don't Gate

You don't get no output no matter what's at the inputs. It is believed that the don't gate was the breakthrough that made the LSI write only memory possible.

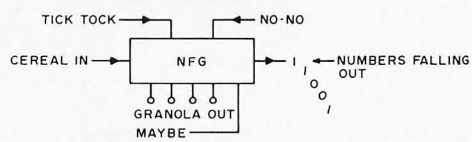
NOISE EMITTING DIODE



Noise Emitting Diode (NED)

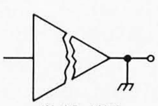
When connected across a 1000 volt supply it makes a loud noise (once). The NED was discovered by Igor Pravaganda whom you'll recall worked many years trying to filter AC with electrolytics. He'll always be remembered as the father of the confetti generator.

SHIFTLESS REGISTER



Shiftless Register

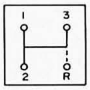
Must be used with 3 speed forward clutch gate. Shifts at 15, 25, and 35 bits per second. Double clutching with logic 2s is not suggested.



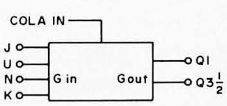
IN-OP AMP

Inoperational Amplifier (IN-OP AMP)

Linear cousin of the DON'T gate. Provides no output for any input at a slew rate of 0 volts per microsecond. Mil Spec. version available at 100 times the cost of OEM version.



CLUTCH GATE



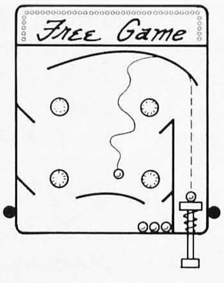
J (UN)K FLIPFLOP

J(UN)K Flip Flop

Doesn't change state when clocked regardless of input states. Changes state only when cola machine down the hall makes change.

Excess 3 To Insufficient 4 Carry Forward Fudger

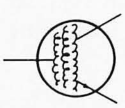
Used to enter Murphy factor and get the programmer off the hook.



EXCESS 3 TO INSUFFICIENT 4 CARRY FORWARD FUDGER

Moss

Highly experimental material. Very rare at present since only source is from under grizzly bear toenails. Turns green when facing north while on wood substrate.



MOSS

Fuzz Locked Loop

Great if you want to avoid radar speed traps. ■



The New ALTAIR 680

James B Vice
MITS Inc

The new ALTAIR 680 designed by MITS is a system based on the 6800 microprocessing unit (MPU). The MPU is available from Motorola or American Micro-Systems and adapts nicely to a minimum design configuration.

The ALTAIR 680 case measures about 11" by 11" by 4-3/4" (28 cm by 28 cm by 12 cm) making it less than one third the size of the ALTAIR 8800. The basic system is available in three configurations, depending on the intended application. These include a user programmable processor with complete front panel controls, and two smaller versions oriented towards dedicated ROM programmed applications.

The compact size of the 680 obviously precludes any significant amount of internal expansion, although additional memory and IO control are already on the drawing board. Its small physical size can be deceiving. The overall concept was to keep the machine as simple, small and inexpensive as possible; but it forms the complete central processor of a system in itself. All that is needed to make a MITS 680 system is the addition of some IO devices and software.

The Three Models

The construction of this machine is a relatively easy matter for even the most inexperienced kit builder. Almost all of the circuitry is contained on a single large printed circuit board, including memory and a built-in IO port. This single board is a full central processor with the exception of a power transformer and some control switches. This is where the main distinction between the three configurations is encountered:

- Most hobbyists will be concerned with the full front panel model. This contains all of the necessary controls for addressing and entering data besides those for controlling the processor itself.
- A turn key front panel model is also available which eliminates all controls except restarting the processor's ROM software. This could be used in applications where it is desirable to eliminate the possibility of the operator or any other person affecting the machine's memory or computing cycle. An example for such an application might be its use in controlling an intrusion detection system, or for a manufacturing machine control system.
- The third configuration is similar to the turn key version. The 680 will also be available as just the large PC board mentioned above. This board contains

everything but a power supply and controls. Its application is similar to the turn key model, except that the computer would be "buried" inside another machine.

The board only model is an excellent starter for the experimenter who wishes to purchase an absolute minimum and do a bit of his own designing. Such experimental use is aided by the considerable amount of information available on the 6800 microprocessing unit from Motorola Semiconductor Products, Inc. The 6800 MPU is also TTL compatible and requires only one 5 volt power supply.

Front Panel

In the front panel model of the 680 there is an additional printed circuit board. This board contains all of the logic circuitry necessary to reset, halt or start the processor. Also located on this board are switches and associated LED indicator lights for each of the sixteen address lines and eight data lines. The front panel printed circuit board mounts directly to the main printed circuit board via a 100 contact edge connector. This eliminates the need for a cumbersome wiring harness. The only other control is the power switch, located on the back panel of the unit for safety purposes.

On the dedicated program models, no front panel is needed because PROM or ROM software is used to store the starting address; a minimum fixed set of programs must be supplied by the user or manufacturer in this form of the system.

Functional Description

The basic ALTAIR 680 computer can be subdivided into five functional sections. These are the MPU and clock, the memory, an IO port, control and indication, and the power supply.

The first three of these sections, along with the power supply regulation components, are located on the main printed circuit board.

MPU and Clock

At the heart of the 680 system is the 6800 microprocessing integrated circuit. This is a versatile and very powerful little processor, yet it is directly responsible for the overall simplicity of the 680 design.

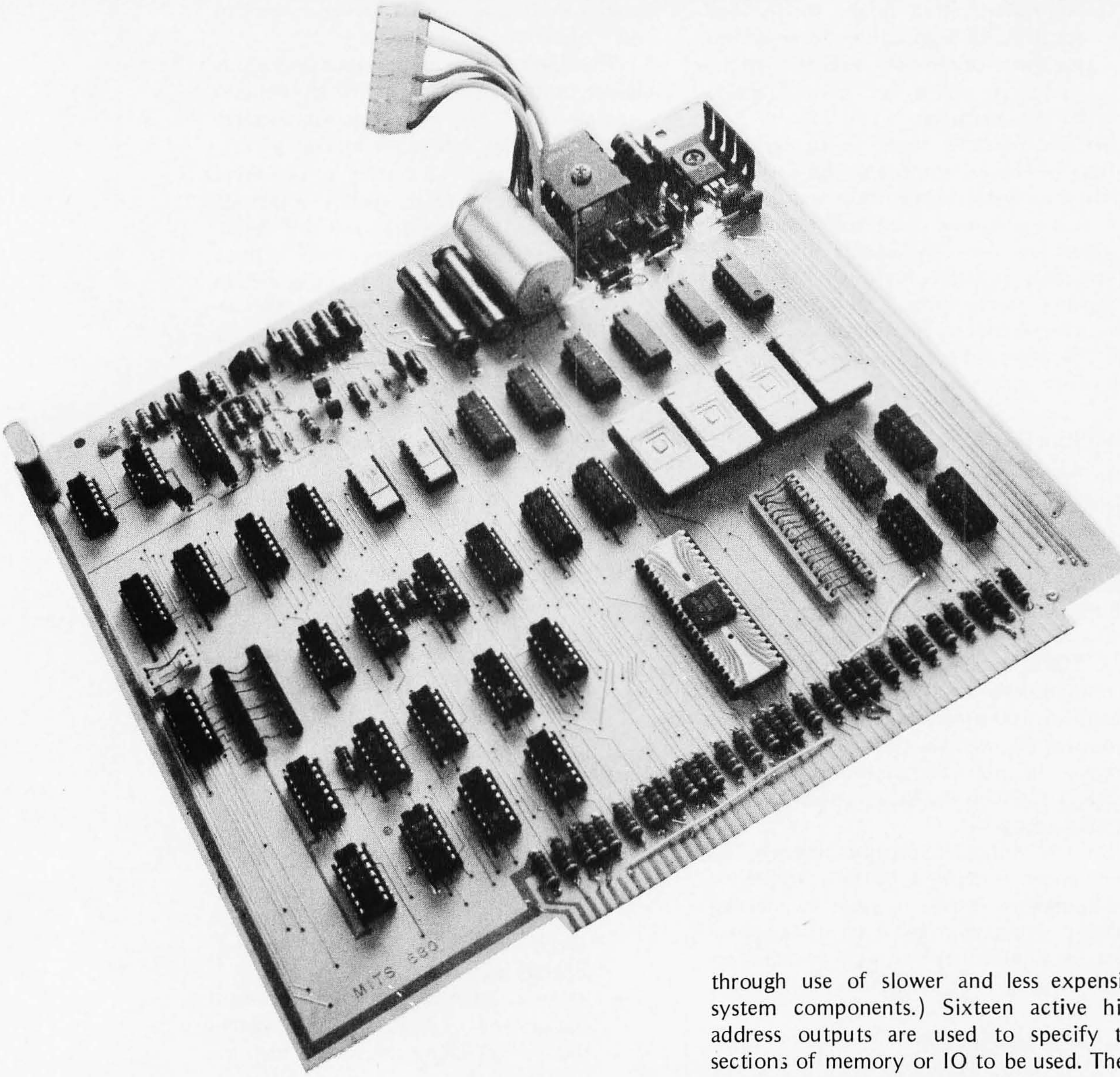
The 6800 is an 8 bit parallel processor using a bi-directional data bus and a 16 bit address bus. The address bus gives it the ability to directly address 65,536 bytes of memory. (Of course most configurations will have fewer than 65,536 bytes.) The instruction set consists of 72 basic instructions with various addressing modes giving a total of 197 different operation codes.

The 6800 has seven different addressing modes, with the available modes being a function of the type of instruction selected. The seven modes include the following:

- Accumulator Addressing – one byte instructions which specify either of the two accumulators use this mode.
- Immediate Addressing – two or three byte instructions with data specified in the instruction use this mode. In immediate mode instructions, one or two bytes of data follow the op code, depending upon the instruction involved.
- Direct Addressing – two byte instructions which allow the user to directly address the first 256 bytes of memory address space in the machine employ this mode.
- Extended Addressing – three byte instructions with a full 16 bit address in the second two bytes use this mode. There is no need to set up an on-chip register to access all of memory with the 6800. This mode is available for most data manipulation operations.
- Indexed Addressing – two byte instructions with this mode add the second byte of the instruction to the 16 bit index register to give the address of the operand.
- Inherent Addressing – certain one byte instructions imply the operands directly and thus do not need a separate address.
- Relative Addressing – all the branch instructions calculate the branch address by adding the second instruction byte to the current program counter plus two. The relative offset is treated as a signed two's complement number (8 bits) being added to the address in the program counter. This allows the user to branch to memory location +129 to –125 bytes from the location of the present instruction.

These various addressing modes may take a bit of getting used to, but once understood they allow for some very fast programs to be written.

The 6800 MPU contains three 16 bit registers and three 8 bit registers. The program counter is a two byte register which keeps track of the current address of the program. The stack pointer is also a two byte register which contains the next address in a variable length stack found in main memory. The index register is a two byte register used to store data or a memory address for indexed addressing operations. There are two single byte accumulators used



for holding operands and results from the arithmetic logic unit (ALU). The 8 bit condition code register indicates the results of an ALU operation. In this register there are two unused bits, kept at a logic one. The remaining six bits are used as the status flags for carry, overflow, zero, negative, interrupt and half carry.

There are several timing and control signals required to operate the MPU. Two clock inputs are required, phase 1 and phase 2. These must be nonoverlapping and run at the Vcc voltage level. Ordinary TTL will not drive these clocks properly. In the 680 the clock is a 2 MHz crystal controlled oscillator with logic to provide a 500 kHz two phase clock. (Although the 6800 is capable of running with a clock of up to a 1.0 MHz, MITS has set the speed of the 680 to 500 kHz in order to greatly reduce the cost through use of slower and less expensive system components.) Sixteen active high address outputs are used to specify the sections of memory or IO to be used. These can drive up to one standard TTL load and 130 pF. There are also eight bi-directional data lines with the same drive capability as the address lines. The $\overline{\text{HALT}}$ signal is an active low input which ceases activity in the computer. The RW (read or write) signal in the high state indicates that the processor is in a read condition; in the low state it indicates that the processor is in a write condition. The VMA (valid memory address) signal tells external devices that the processor has a valid address on the memory bus. The DBE (data bus enable) signal is the input which enables the bus drivers. The BA (bus available) signal indicates that the machine has stopped and that the address bus is available. $\overline{\text{RESET}}$ is used to reset and start the MPU from a power off condition. The $\overline{\text{IRQ}}$ (interrupt request) signal, when low, tells the processor to start an interrupt sequence. This can occur only if the interrupt mask bit in the condition code register is low. The $\overline{\text{NMI}}$ (nonmaskable interrupt)

signal is essentially the same as the $\overline{IRQ}$ signal except that it is not dependent on the condition code register.

Memory

The main printed circuit board on the 680 contains the basic memory for the unit also. This includes 1024 bytes of random access memory and provisions for another 1024 bytes of read only memory. The random access memory circuits being used are the 2102 static 1024 X 1 bit parts. Read only memories of the mask programmed type can be custom ordered, and are very expensive in small quantities. The 1702 type, ultra-violet erasable programmable read only memories are typically used in this system. These are 256 X 8 bit units, so four 1702As would be required to fill up the available space in the 680.

There is additional memory for the 680 on the drawing board at this time which may add up to 12 K bytes more storage to the unit.

IO Port

Also on the main printed circuit board is a built in IO port and the appropriate interface circuitry. This port may be configured as either an RS232 level port or either a 20 mA or 60 mA current loop TTY level port. This means it can be interfaced with proper software to the old Baudot type Teletypes, such as the Model 19 and Model 28 machines.

The entire design of the 680 is greatly simplified due to the 6800's use of memory address space for IO addressing. The processor uses addresses to refer to IO devices as well as memory, rather than have special IO instructions and a separate IO bus. Within the limits of practical engineering, programming and memory requirements, as many IO devices as desired can be added to a 6800 microprocessor system. No logical limitation is built into the instruction set. MITS also has additional IO interfaces on the drawing board at this time; although availability of this and the additional memory boards will be greatly influenced in their development by customer response.

Control and Indication

On the fully user-programmable version of the 680, the front panel assembly contains a RUN/HALT switch with an LED indicator for each switch position. There is a RESET switch with no indicator, and another indicator for the AC power switch which is located on the back panel of the unit. The switches for the 16 address lines and 8 data lines, and their associated indicator lights, are also located on the front panel assembly of the fully programmable model. There is also a DEPOSIT switch.

The DEPOSIT, RESET, DATA and ADDRESS switches are enabled only when the RUN/HALT switch is in the HALT position. To view the data in a particular memory address, the RUN/HALT switch must first be in the HALT position and then the ADDRESS switches may be set to the required address. The data located at that particular address will then appear on the DATA LED indicators above the DATA switches.

To write data in a desired location, once the correct address has been set on the address switches, the appropriate data should be entered on the DATA switches and then the DEPOSIT switch activated. Since the address bus is already connected to the switches by being in the HALT state, a write pulse causes the data to be written into the selected RAM address.

When the RESET switch is activated, the processor itself resets. This initiates a restart sequence, pulling the address bus to its high state and causing hard-wired data on the board to be used as the restart address.

On the dedicated program versions of the 680, most of these functions are taken care of by ROM or PROM. The only controls available to the user are the AC power and RESET switches.

Power Supply

The 5 volt supply to the computer is supplied from the power transformer through a conventional bridge rectifier and filter capacitors and voltage regulator IC. A 32 volt winding on the transformer is used to generate the unregulated ±16 volts required for a TTY interface, and a −16 volt line is fed to four zener diode regulated outputs to provide four −9 volt lines for the PROMs.

The transformer itself, along with the power switch, is located on the computer's back panel. There are also provisions for installing a cooling fan when necessary.

As far as software goes, MITS has a package available similar to the 8800's Package One. This includes an editor, PROM monitor and assembler. This all goes to make the ALTAIR 680 a rather powerful little machine. There is also the possibility for further software development.

MITS has decided to await customer response to determine the course of further 680 development in both the areas of software and hardware.

Although it's not quite as powerful as the ALTAIR 8800, the ALTAIR 680 is mighty close and costs less.■

How to Save the BYTEs

Thomas C McIntire
310 Village Dr
Centerville OH 45459

The personal computer is today, a real, affordable entity. A paradox is appreciated, however, when configuring a system. You can order a computer for less than a good color television. But talking to it costs much more. Computer communications, meaning the traditional twins of input and output, are commonly referred to as IO. The paradox is that it costs more to save the bytes than to process them.

A friend has recently completed assembly of an ALTAIR 8800 and is interfacing a TV typewriter and an audio cassette recorder for bulk storage. The keyboard, TV, cassette recorder and electronics cost more than the computer!

Another popular approach is to use a tele-typewriter. Combining the keyboard, serial printer and punched paper tape IO into one machine, you might get by with as little(?) as one K dollars.

A traditionalist, looking back a couple of generations, might opt for a key punch of maybe 026 vintage. Assuming this is affordable, a card reader must also be had. Still, no back talk. You have input, but no output. Even the veteran of the tab card shop doesn't enjoy reading the Hollerith coded holes.

Magnetic recording media is the present vogue of the big boys. Random access storage on disk has many virtues. Unfortunately, disk drives and Cadillacs have a common trait ... price. The Pinto-like affordability of the cassette devices is a partial solution, but printed output is still desired. Punched paper tape could be dual purpose, but you might end up as cross-eyed as the tab card interpreter.

Looking at the ads, I find a Centronics printer might cost about the same as a three year old Ford ... but, my wife says I can't sell her car.

Since the ENIAC, man has been busy devising various ways to convert the printed word to a form recognizable by a machine. And, with the number crunching completed, the machine has to output the results in a form that is readable by man. Some specialized attempts to make character representations dual purpose have been very successful. Notice the MICR encoding near the bottom of your personal checks. Notice also, the precision printing and magnetic recognition units involved are usually owned by those who can afford it . . . the banks.

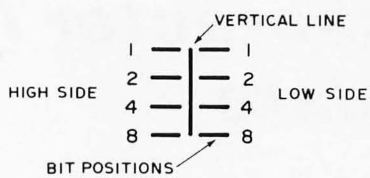
Because I don't own a bank (the converse is more nearly correct), and a Cadillac isn't in my home-owned computer budget, a cheaper method is sought for hauling and storing my labors-of-love bit strings. Building my own printer might have some merit. There are many proven architectures from which to choose. Ball, basket type, drum, belt, moving head, move the paper, move both, are some of the many possible. All commercial printers have one thing in common . . . a proliferation of moving mechanical parts. The major reason is our alphabet. If we changed the character set, printing would be simpler. If a semblance of logic prevailed, optical recognition might be easier.

The puritanical school of thought is to require as little change in man's ways as possible when dictating computerizing conventions. The backyard bit fiddler is usually more condescending, and I for one wouldn't mind bending the rules a little.

Applying a true structured programming approach: first, the character set; we'll get on to the printing and reading later. The ASCII code representations are both logical and suit computer oriented, binary storage nicely. This fact may well have occurred to the originators: In hexadecimal notation the letter A is 41; the letter B is 42; and the letter C is 43, etc. What is needed is a way to encode in a given printed character the entire byte range of 00 to FF. If we use a

single, vertical line as the root of a character, visual focusing (and machine registration) can be achieved. The left and right side of the line, respectively, can denote the HIGH and LOW significance. Top to bottom we assign positional values of 1, 2, 4, and 8. Where 1 is the top and 8 is the bottom, 2 and 4 are equidistant between. Using the text book descriptions of ON and OFF, we can draw connecting, horizontal lines to code the bits.

The coded print value of hexadecimal FF is thus:





Such symmetry! Hercule Poirot, the Belgian detective in Agatha Christie's novels, would surely applaud. Using this scheme, we can encode the entire range of bits possible within a byte. The chart that follows is a partially encoded ASCII code set.

As an experiment to test the "human" readability of such graphics, I selected an unbiased sampling group . . . my kids. The subjects: Sons, ages 8, 10 and 11. Our 17 year old daughter also participated to appease women's lib. Motivation was to learn the alphabet and I would pay one dollar. Total research and development costs: $4.

The method: I used blank 96 column cards and drew one character per card, encoded as above, with the conventional letters printed on the backs. Each participant studied the flash cards at their own leisure. Intensity and frequency of study was dictated by individual need and greed. NOTE: A dollar seems like more to an eight year old than to a teenager. Group study was suggested, but resulted in some disharmony at times.

The results: After one week of elapsed time (and curtailed allowances) all four subjects successfully memorized the coded character set. Testing was conducted by myself and critiqued by my wife. In turn, for each examinee, I flashed the cards in random order for their response. Individual results were as follows. Eight year old, three mistakes, corrected with prompting. Our number two son had only two mistakes, self-recognized and corrected without prompting. The oldest boy needed the dollar . . . no mistakes. Not too impressed with the funding, our teenager didn't make any mistakes, but considerable prompting was necessary.

Conclusions: A dollar doesn't buy much. Kids have good memories. This scheme should work well in junior high schools. Critic's comment: "Try a re-test, but with the cards turned around." The test group also unanimously suggested that I attempt the test myself . . . overruled . . . rank has its privileges.

The problem of printing and reading such a character set was broached next. To keep mechanical movements to the minimum, the plan is for using friction-fed, rolled paper tape imprinted by a matrix type electromechanical assembly. Reading is by detection of intensity of reflected light from a common source, carried from the read head by optical fibres. My current efforts are aimed at scrounging and evaluating surplus and junk parts for fabrication.

It is highly doubtful that this scheme will revolutionize the industry, but hopefully, I can save my bytes inexpensively . . . and teach my children hieroglyphics.■

NEW! ... VDM-1

features-

- *ultra high speed intelligent display*
- *generates 16, 64 character lines of alpha-numeric data*
- *displays upper and lower case characters*
- *full 128 ascii characters*
- *single printed circuit card*
- *standard video output*

- $160.00 -

SPECIAL FREE OFFER!

Scientific Notation Software Package with Formatted Output

The floating point math package features 12 decimal digits with exponents from +127 to −127; handles assigned and unassigned humbers. With it is a 5 function calculator package: + − X ÷ & sq. root. It includes 3 storage and 3 operating memories and will handle chain and column calculations.

With the purchase of (1) VDM-1 and (1) 4KRA-4 Memory:

Just $299.00 (Offer expires 2-1-76)

from-

Processor Technology

More on the SWTPC 6800 System

Gary Kay
Southwest Technical Products Corp
219 W Rhapsody
San Antonio TX 78216

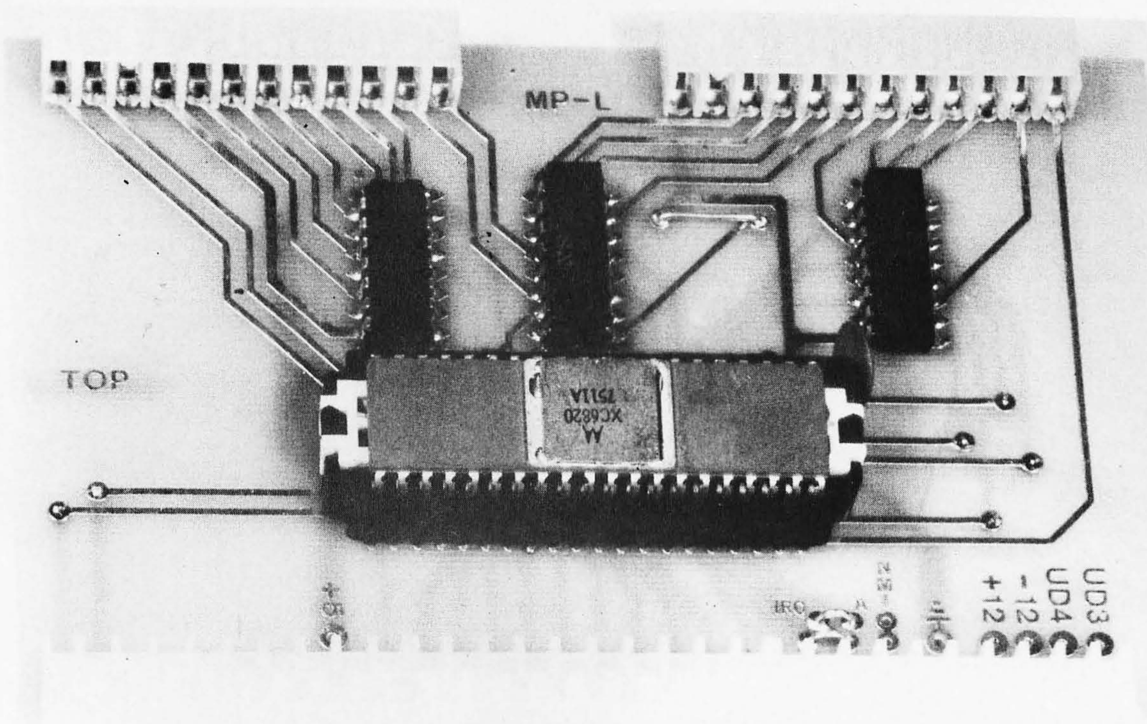
The Parallel Interface Board is used to latch and control the input and output of 8 bit bytes. The Motorola 6820 Peripheral Interface Adaptor (PIA) is the main component of this board, with several smaller chips acting as buffers. This board permits parallel connections to such devices as printers, laboratory breadboards, and special purpose keyboards.

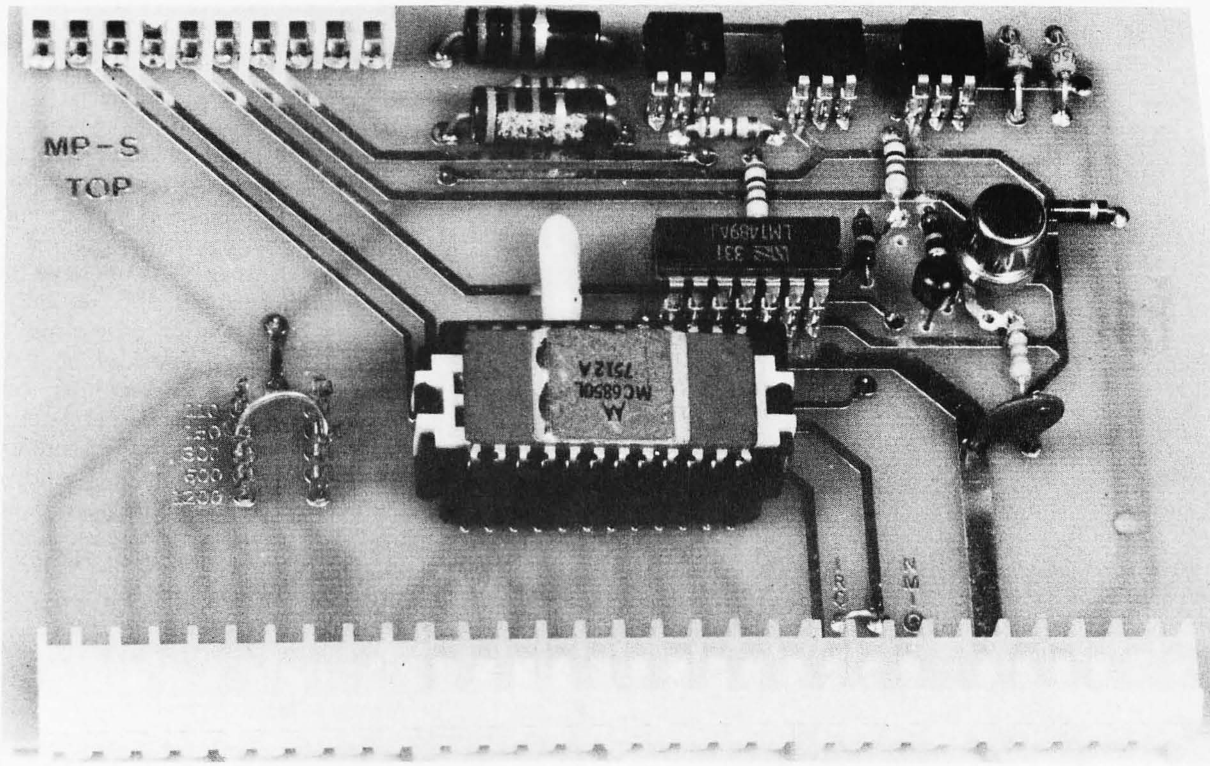
In the December 1975 issue of BYTE we talked about the microprocessor/system board (MP-A) and the mother board (MP-B) for the Southwest Technical Products 6800 microprocessor system. This article describes the serial control interface (MP-C), the 2,048 byte random access memory board (MP-M), the power supply (MP-P) and cabinet (MP-F). We will also talk about the serial interface boards (MP-S) and parallel interface boards (MP-L).

Serial Control Interface

The serial control interface (coded MP-C) is a 5.25 inch by 3.5 inch doubled sided, plated through hole board containing a 6820 peripheral interface adapter integrated circuit and circuitry which forms the serial control interface. Data rates of 110 or 300 baud are selected by a jumper wire. The interface includes software control of an input to output echo feature which is necessary in some tape reader operations. Its data input or output must be in ASCII (without parity) and either 20 mA Teletype or RS-232 compatible. A low cost terminal such as the TV Typewriter II (February 1975 issue of Radio Electronics Magazine) is ideal.

IO connections are made to the interface via a 10 pin connector along the top edge of the board. Power for the board is provided by a 5 VDC regulator at a current consumption of about 0.2 A. 12 VDC and −12 VDC sources are also used.





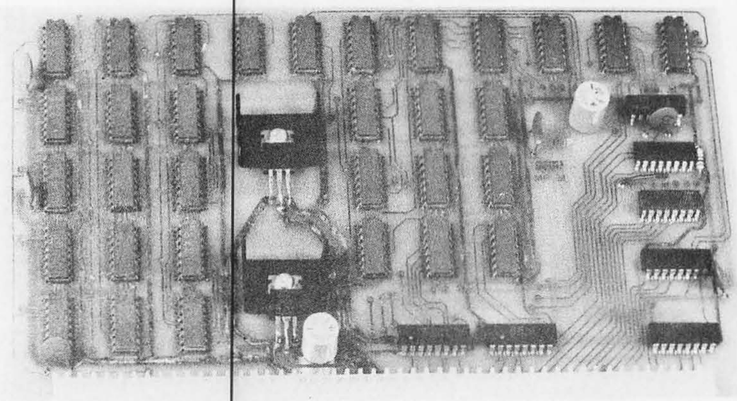
The Serial Interface Board is used to convert parallel data from the processor into serial data to a terminal (and vice versa). The major portion of this task is accomplished using the Motorola 6850 Asynchronous Communications Interface Adapter (ACIA) circuit, which is the large package in the center of the board. The remaining components on this board are used to provide both an RS-232 interface and a Teletype 20 mA current loop interface.

The board itself must only be plugged onto the first interface port position of the mother board. It is constantly polled for incoming commands by the Motorola MIKBUG software stored in the operating system ROM whenever the system is powered up, or is reset and is under operating system control. When system control is turned over to your program, the control terminal is also available for program IO. To output a character to the terminal's display, store the character in accumulator A and jump to subroutine OUTEEE, which is a character output routine written into the operating system ROM. To input a character from the control terminal's keyboard, jump to subroutine INEEE, which is a character input routine written into the operating system ROM. In this subroutine the system hangs in a loop until a character is typed at which time there is a return from subroutine with the entered character deposited in accumulator A. The use of these ROM stored subroutines greatly simplifies the job of the programmer for control terminal data input/output.

In addition to the Serial Control Interface, any combination of up to seven parallel or serial interfaces may be plugged onto the interface connectors. Since the 6800 family of chips includes both parallel (6820) and serial (6850) interface elements, interfacing is extremely flexible.

Parallel Interface

The Parallel Interface (coded MP-L) is a 5.25 inches X 3.5 inches (12.86 cm X 8.57 cm) double sided, plated through hole circuit board containing a 6820 peripheral interface adapter integrated circuit and its associated circuitry which is used to connect a parallel data device such as a printer or parallel data terminal to the computer system. The board is provided with two separate connectors along the top edge of the board. One has eight fully buffered TTL compatible high current data outputs along with one buffered "data ready" output line and one "data accepted" input line for complete handshake control. The other connector has eight TTL compatible input lines along with one "data ready" input line and one "data accepted" output line, here again for complete handshake control. The "data



The memory board, shown here with a full 4096 word complement of 2102 memory chips, is one of the more important elements of the system. The black chips at the right edge of the photo are interface devices and address decoding. The two voltage regulators on the board are in the center. The remaining integrated circuits are 32 chips of 1 K by 1 bit memory.

ready" and "data accepted" lines are under complete program control even to the extent of setting the transition polarity upon which the lines will be triggered. Interrupts are under complete software control as well.

For the user who has specialized parallel IO requirements, the TTL data buffers may be omitted from the board, and each of the sixteen data lines may be individually software programmed by the user as either all inputs, all outputs or any combination of the two. The programmer has complete software control of the four handshake lines, two of which are software programmable for input or output. Power for the board is supplied by a 5 V regulator at a current consumption of 0.3 A.

Serial Interface

The Serial Interface (coded MP-S) is a 5.25 inches X 3.5 inches (13.3 cm X 8.9 cm) double sided, plated through hole circuit board containing a 6850 asynchronous communications adapter integrated circuit and its associated circuitry which is used to interface a serial device such as a terminal to the computer system. Like the Serial Control Interface, its communication must be in ASCII form and either 20 mA TTY or RS-232 compatible. Baudot coded teletypes will not work. The data IO baud rate for each of the interfaces is jumper programmable and may be set for 110, 150, 300, 600 or 1200 baud operation. One central clock on the microprocessor/system board provides all of the various baud rate clocks simultaneously, so that each of the serial interfaces can have an independent data rate. This eliminates a good deal of duplicate circuitry and keeps the serial interface cost low.

As with the Parallel Interface, there are many functions that are under software control. Selection of one of 8 different combinations of bit count, parity, and number of stop bits is user programmable as is control of transmitter and/or receiver interrupts. Checking the interface for transmitter buffer empty, receiver buffer full, framing error, parity error, and receiver overrun are here again all done through software just by reading the data contained within the interface's internal status register. External connections to the board are made via a ten pin connector along the top edge of the board. Power for the board is supplied by a +5 V regulator at a current consumption of approximately 0.2 A. +12 VDC and −12 VDC sources are used for generating the Teletype currents and the RS-232 voltage output.

Memory Board

The Memory Board (coded MP-M) is a 5.5 inch by 9 inch (14 cm by 22.9 cm) double sided plated through hole circuit board with data bus buffering, and address decoding for up to 4,096 bytes of fast 2102 static random access memories. The basic memory board kit comes with only 2,048 words, however. To fill the board to a full 4,096 words of RAM, you must add the memory expansion kit (MP-MX) which contains another 2,048 words of memory ICs and a separate voltage regulator. Up to four of these 4,096 word boards may be plugged onto each mother board.

The 2102 static memories were chosen because of their availability, low cost and established reliability. Although the 4 K dynamic memories are becoming popular, they require refresh circuitry and slow the processor during refresh cycles. Address assignments are made on each memory board by connecting the address jumper to one of the eight possible positions, progressing on each memory board from 0 to 7. This programs the boards from 0 to 32 K words in 4 K word increments. Since each mother board will only support up to four 4,096 word memory boards, it is necessary to use another mother board with separate power supply to expand the memory beyond 16,384 words. Power for the lower 2,048 words of memory as well as the decode and buffer circuits is provided by a 5 V regulator with a current consumption of approximately 0.8 A. Power for the upper 2,048 words of memory when present is

provided by a separate 5 V regulator at a current consumption of approximately 0.6 A.

Power Supply

The MP-P power supply consists of a power transformer, high current bridge rectifier, filter capacitor, and power supply board. The low voltage transformer secondary winding, bridge rectifier and filter capacitor provide the 7 to 8 V DC at 10 A required by the complement of boards in the computer system. Since the regulation down to 5 V is provided on each of the system boards, the actual value of this voltage is not critical. It must however be maintained at no less than 7 V for proper regulator operation while not so high as to cause the regulators to generate abnormal temperatures.

The higher voltage transformer secondary winding along with the rectifiers and filter capacitors on the power supply board provide the +12 and −12 V DC at 0.5 A required by the control and serial interfaces. All connections from the power supply to the mother board are made through an easily detached connector on the power supply board. This makes mother board installation and removal a snap. The power transformer's primary may be wired for either 120 or 240 VAC operation with a current consumption of 120 VAC at 1 A or 240 VAC at 0.5 A.

Chassis and Cover

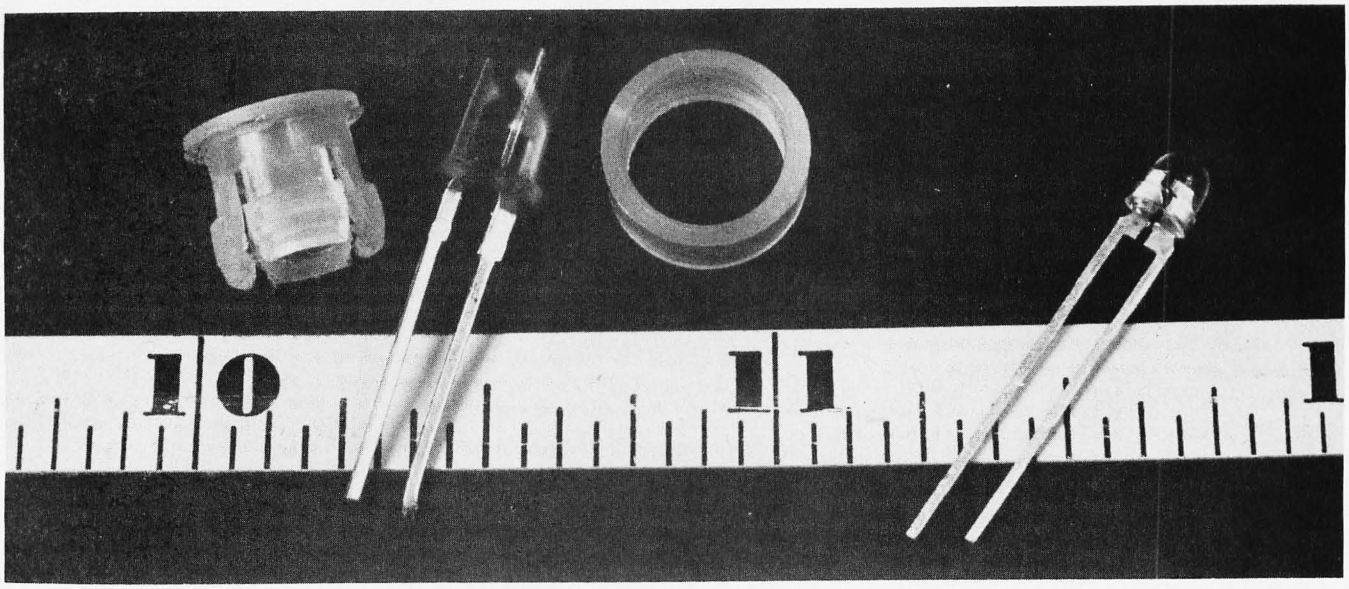
All of the boards for the 6800 computer system including the power supply are housed in a 15.125 inches wide X 7.0 inches high X 15.25 inches deep (37.05 cm wide X 17.15 cm high X 37.36 cm deep) anodized aluminum chassis with a perforated cover. The use of the perforated cover eliminates the need for a cooling fan in almost all environments. The front panel supports both the POWER on-off and RESET switches. The RESET switch initializes all of the registers in the system and loads the terminal controlled Motorola MIKBUG operating system whenever depressed. The rear panel contains an array of holes through which the interface cables and line cord may pass. Both panels along with the cover may be easily removed providing 360° access to the system for prototyping or service.

The 6800 system presented within this and the previous article, has been shown to have outstanding ease of use and is an economical package. But as many of us already know, hardware is but a small part of a "computer system." Programming, or software as it is generally referred to, is just as important as the hardware. Of course this system does have a very useful ROM stored operating system, but what else is available, and how does one load such software in memory without having to type it in through the control terminal one byte at a time? Well, first of all several diagnostic listings are provided by the manufacturer of the kit to help check out the various boards within the system. These diagnostics are typically less than 90 bytes in length and can be entered manually from the control terminal in less than five minutes. Included within these diagnostics are two programs that provide a thorough checkout of the random access memory boards, a common failure point for many systems.

Regarding some method of storing and loading in programs, a low cost audio cassette tape digital storage system is presently in the works that will be totally compatible with this computer system. You can expect to see it in a forthcoming BYTE Magazine article. Also to be available shortly is an editor/assembler software package which will be sold for the cost of the documentation and tape only to those people simultaneously purchasing 4 K of the 8 K words of memory necessary to support the package.

Another note of importance is that the ROM stored mini-operating system on the Microprocessor/System Board is exactly the same (MC6830L7) as that used on Motorola's Evaluation Module and Integrated Circuit Evaluation Kit. This means that most all programs written for the Motorola's Evaluation Module will function on the 6800 computer system presented here. Motorola also supports their more sophisticated prototyping system called the EXORcisor® (Registered trademark of Motorola Inc.). This system has a larger, more sophisticated firmware package, but it uses the same 6800 microprocessor element, therefore much of its software is compatible with the 6800 system presented here. Because of this compatibility, arrangements have been made with Motorola Inc. to allow Southwest Technical Products 6800 Computer System customers to have access to Motorola's 6800 program library. Customers will be permitted to join by either submitting an acceptable program or by paying a membership fee. Either makes them a member of the Motorola 6800 User's Group for two years with access to programs within the library plus upcoming program additions.

For those applications requiring the utmost in speed and storage capability, arrangements are in the works with ICOM Corporation to supply a floppy disk and floppy disk operating system (FDOS) that is compatible with the 6800 system described in this article. ■



This photograph illustrates two common LEDs which were purchased from suppliers advertising in the pages of BYTE. The grouping at the left is a highly desirable MV5024 lamp with mounting ring and collar. This facilitates a convenient panel mounting of the lamp. The lamp at the right is a Hewlett Packard 5082-4487 LED which is typical of the small LEDs which are useful as in-circuit indicators.

LEDs Light Up Your Logic

E W Gray
3321 O'Henry Dr
Garland TX 75042

Of the many technological developments in recent years, perhaps none is more intriguing than the visible light emitting diode, or LED. This tiny semiconductor device, with its hypnotic red glow, is a far cry from the cat whiskered crystals which introduced so many of us old-timers to the wonderful world of electronics. A crystalline compound of gallium, arsenic, and phosphorous, the LED is a diode that emits visible light when forward biased. No attempt is made here to explain how the light emission works. Instead, this article shows how LEDs can be used to advantage as data displays, fault, and status indicators.

These highly reliable devices have largely eliminated burnout associated with incandescent indicators. Their small size, low cost, and low power requirement make their use as indicators in circuits very desirable. The LED is ideally suited to today's digital circuits, since it can be driven directly from the logic elements.

Some Applications

Hand-held logic probes, many of them using LEDs as indicators, are well known (see December 1975 BYTE 4 for Alex. F. Burr's review of several commercial probes on page 20, and for Bob Baker and John Errico's plan of a handy logic test clip using LEDs as readouts on page 26). But why not simply incorporate the diode directly into your next project? Select the circuit points which reveal the most about circuit operation and monitor them with LEDs. For example, a single LED driven from a keyboard strobe signal monitors the data transfer operation each time a key is depressed.

Or devote a diode to the output of the system clock generator. Given a clock rate, select the diode's current limiting resistor so that the diode glows dimly when the clock is operating normally. Should the clock fail, the LED will either be extinguished or at full brilliance depending on which logical state results from the failure.

Obvious applications include the outputs of control flip flops and state counters. Not so obvious, perhaps, is a shift register output monitored to show normal data recirculation, or diode logic built with LEDs as a training aid.

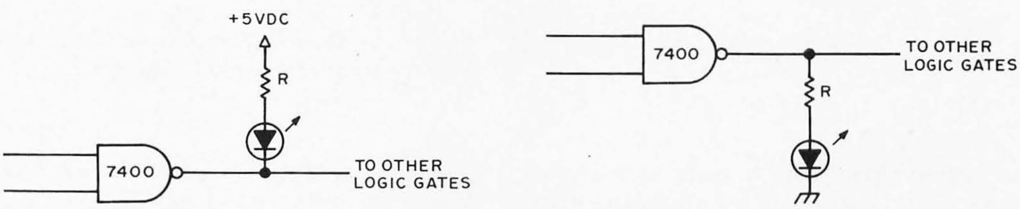


Figure 1: In circuit indicators. A: The diode lights when both inputs are at the high level.

B: The diode lights when either input is at the low level.

LED = Light Emitting Diode is a specially constructed diode which emits light when forward biased.

In a digital system having several registers, the addition of a multiplexer provides a path to switch selected register content to a LED display. In a bus oriented system, use a bus receiver to drive LEDs from the bus itself (one diode for each bit) and all data sources in the system are available for visual examination.

Designing With LEDs

Rules for using light emitting diodes are simple. As with any semiconductor device, read the specification sheet *first*. The typical spec sheet will contain the device description, dimensions of the package and mounting details, and perhaps a few device features or applications. Most important, however, is the section defining electrical characteristics. It is from these specifications that we determine how a particular device is to be operated. For most designs, the only information required to insure satisfactory operation will be the diode's forward potential (V) and forward current (A) ratings. These parameters allow selection of the series current limiting resistor used to control brightness. Relative brightness of the emitted light is dependent on the forward current. Generally, a diode will begin to emit light at about .5 mA forward current. The light is easily seen at about 1 mA to 5 mA. Forward currents of 10 mA to 20 mA bring most diodes up to full brilliance. Most manufacturers include a graph or chart in their specification sheet which illustrates brightness versus forward current. 20 mA is usually the recommended value for viewing over any distance.

Having chosen the desired brilliance and the required forward current, and given a specific supply potential (S), the current limiting resistor (R), which is to be wired in series with the LED, is calculated according to the following formula:

$$R = (S - V) / A,$$

where S and V represent the supply and forward potentials, respectively, which are measured in volts, and where A stands for the forward current measured in amperes. Note that one milliampere equals a thousandth of an ampere (.001 A = 1 mA).

As an example, suppose it is desired to operate a LED at 10 mA forward current. The diode will be driven from a transistor-transistor logic (TTL NAND gate. The power source is rated at 5 VDC. The LED specification sheet requires a forward potential of 1.65 V. The calculation is:

$$R = (5 - 1.65) / .01 = 335$$

The closest standard resistor of 330 O can be used.

As with any semiconductor device, read the specification sheet first.

Some Typical Circuits

Two methods of monitoring TTL logic using permanently installed circuit indicators are shown in figure 1. In part A of figure 1 the diode will light when the gate output is at the low level. Since the 7400 NAND gate is capable of 16 mA sink current, the diode current should be limited to this amount. Normally, the gate will be used to drive other logic; the driving ability of the gate (fanout) will be reduced by the amount of current diverted through the diode, divided by the current required at a gate's input. For example, TTL gates require 1.6 mA input current at the low level. If the LED is operated at 10 mA, fanout will be reduced

An LED can be driven directly by TTL logic for use as an indicator.

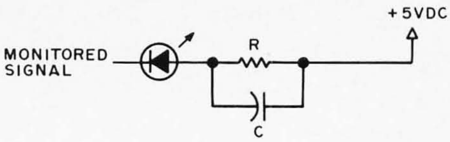


Figure 2: Bypass capacitor increases light level for signals having low duty cycle.

LEDs can be used with logic families other than TTL. Such uses may require buffers and attention to ratings.

LEDs embedded in seven segment displays are no different from discrete LEDs.

by slightly more than 6 loads, so the gate driving the diode should not drive more than three additional gates.

In part B of figure 1, the diode lights when the gate output is high. Although fanout of the driving gate is not reduced by this circuit, the DC noise margin at following gate inputs is reduced. The diode current limiting resistor (R) and internal resistance of the driving gate form a voltage divider which reduces the high level output potential of the driving gate. The diode current must be limited to a level that will not reduce the gate's output level below the specified minimum for a logical one. For the 7400 NAND gate, this is 2.4 V. For reliable circuit operation this minimum must be maintained.

It is often desirable to monitor circuits having a relatively low duty cycle, such as strobe lines to input or output devices. In such cases, adding a bypass capacitor as shown in figure 2 can increase the light level of the indicator. Choose the current limiting resistor to hold diode current at a safe level should the driving circuit fail and remain at the low level.

Using LEDs with DTL or CMOS

If your logic is implemented in a logic family other than TTL, you can still use LEDs as indicators. Diode-transistor logic (DTL) gates work well using the circuit of part A in figure 1. As with TTL, fanout of the driving gate is reduced.

Complementary metal oxide semiconductor (CMOS) logic, popular because of its low power requirement, may also be used as the LED driving element. Several devices are available which will sink 8 mA when used with a 10 to 12 V supply. Among these are the 74C901 and 34049 inverters. By taking advantage of the short circuit output current specification, the 74C00 NAND gate can be used to drive a LED using a 1000 O resistor at 9 V.

Special Purpose LED Devices

Light emitting diodes are available in many forms other than the single lamp. Most of us are familiar with the seven segment displays used in calculators and digital clocks. An interesting application of this device, using the popular MAN1, is the display of binary information as numerical ones and zeros without the use of decoding circuits. Details are shown in figure 3. The b and c segments are connected to be permanently on. Segments a, e, f and d are connected in parallel and driven by the true data bit to be displayed. Thus when the data bit is false (logical zero) a numerical zero appears on the display. When the data bit is true (logical one) segments a, e, f and d are extinguished, leaving the b and c segments displaying numerical one.

In addition to the popular red lamp, LEDs are available which emit green and yellow light. And at least one manufacturer, Monsanto, produces a lamp having both a red diode and a green diode in the same package. This device forms the basis for a unique panel indicator, as shown in figure 4. Here, a single momentary switch is used to both start and stop an operation. The state

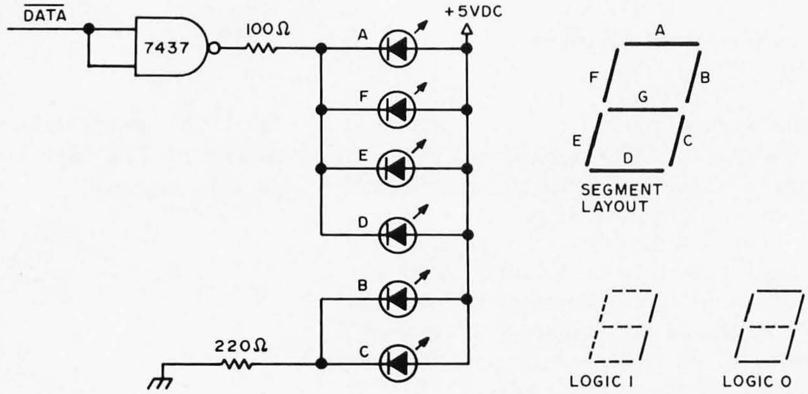


Figure 3: Using a seven segment display to monitor a bit.

of the controlled operation is displayed by the two-color LED. A red light indicates the operation is stopped, while a green light indicates it is running. A typical application would be the start and stop switch on a computer panel. Switch S1 is normally in the center (off) position. A latching circuit composed of two TTL inverters serves as a run and halt flip flop and also debounces the panel switch. Momentary operation of the switch to the stop position sets output A low and output B high. The two-color LED is connected across the latch outputs. When the latch is in the halt state (output B is high) the red LED is forward biased. Resistors R1 and R2 limit the diode current to about 10 mA. Momentarily activating the switch to the start position toggles the latch, back biasing the red diode and forward biasing the green diode. D1 is a 1N914 or equivalent diode which effectively shorts out R2, leaving R1 to limit the green diode's forward current to about 20.0 mA. The different currents are required because the green diode is not as bright as the red diode for the same current.

The same manufacturer also offers a red bipolar lamp suitable for use as an AC line voltage indicator as shown in figure 5. The manufacturer recommends a resistor of 11 kO at 2 W for operation at 110 VAC. ■

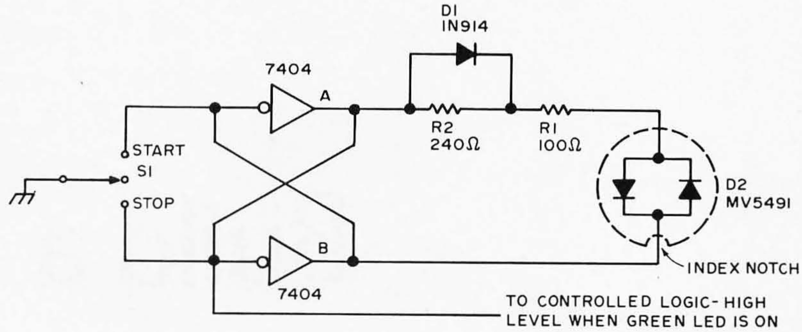


Figure 4: Two-color panel indicator.

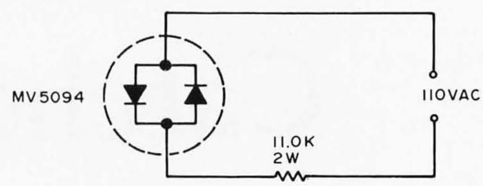


Figure 5: Panel indicator for AC operation.

William A Walde
28 Tuttle Dr
Acton MA 01720

Build a TTL Pulse Catcher

While checking out the operation of some oneshots on the address latch board during the construction of my Mark-8 microprocessor, I discovered that my first homemade logic probe could not detect very short TTL logic pulses. Since my old probe would not work, I needed a quick and easy way to tell whether a short TTL pulse had arrived. I dreamt up this circuit which solved the problem by adding memory in the form of an RS flip flop wired from a NAND gate. This pulse catcher will detect pulses as short as the combined gate delays of the two NAND sections used to form the flip flop, approximately 10 to 30 ns. The circuit works by changing state at the start of a pulse, with an LED monitoring the flip flop output. There is no indication of the end of the pulse and there is no way to tell if multiple pulses have occurred. After a pulse has been detected, the circuit must be reset in order to detect the next pulse. The TTL pulse catcher is designed to work with either a positive or a negative going pulse as selected by a switch.

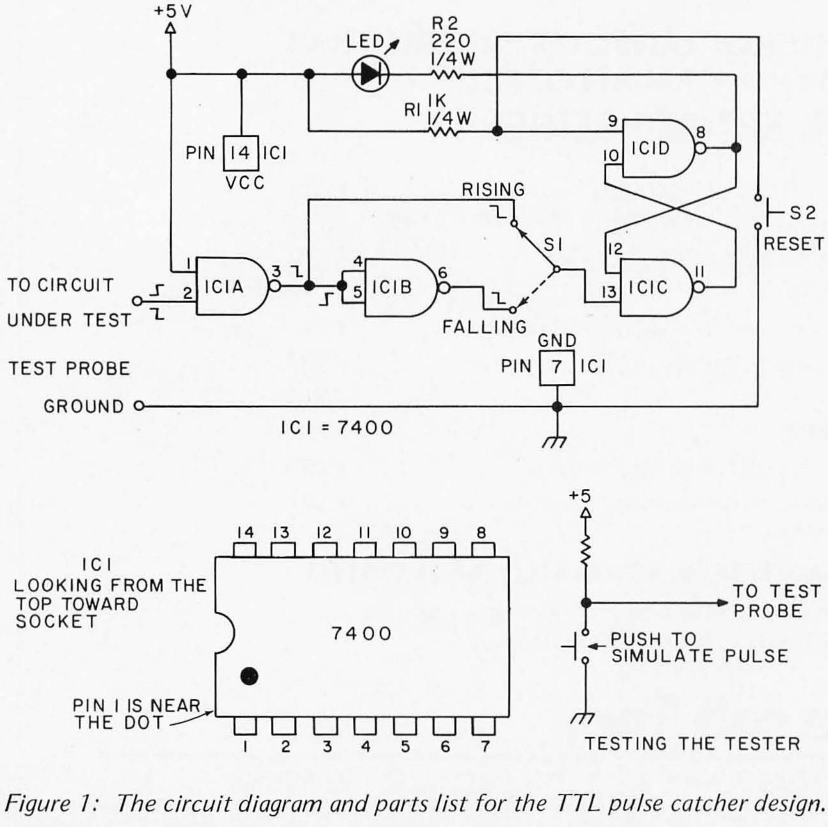


Figure 1: The circuit diagram and parts list for the TTL pulse catcher design.

Construction

The original version of this test instrument was built using perforated board and flea clips. The circuit could also be made using a small printed circuit board. Lead dress is not critical, but the polarity of the LED must be observed. Select a value of the resistor R2 in the range 47 to 470 O such that good illumination is achieved with a current of approximately 20 mA. A typical resistance value is 220 O. The tester can be tested using a resistor, a bounceless pushbutton switch and the test circuit shown in figure 1. Label switch S1 to identify which setting is the rising edge mode and which setting is the falling edge mode. A clever approach would be to build the pulse catcher inside a cylindrical enclosure, such as a pen or thin tube.

Using the Pulse Catcher

Connect the 5 VDC and ground terminals to a suitable power supply. If you use a power supply separate from the main system supply of the computer or logic device you

are testing, be sure to tie the grounds of the two supplies together. Connect the input probe to the line being tested and select the rising or falling edge mode via switch S1. Depress the reset button to extinguish the LED and arm the latch. If you are unable to cause the LED to stay off after releasing the reset, the following information is learned about the line being tested:

It may have a steady state condition which is inconsistent with the mode of the test. Change S1 and try again.

You may be observing a line which has regular clock transitions. The light will never go out when testing such a line.

In normal operation, once the pulse catcher is reset, the LED will remain out until the first pulse comes along, after which it will stay on until reset by pushing the S2 button.

Theory of Operation

Gates IC1c and IC1d are cross-coupled to form the familiar RS flip flop that is used to remember when a pulse has occurred. Momentarily depressing SW2 grounds pin 9 of gate IC1d, causing the flip flop to go into the reset mode. In this mode both sides of LED1 are high and it will not light up. In a similar manner, falling pulse at pin 13 of gate IC1c will cause the flip flop to go into the set mode with gate IC1d now becoming a current sink for LED1, causing it to illuminate.

Switch SW1 is used to select between either a rising or a falling input pulse. The RS flip flop always needs a falling pulse to operate. In the case of a rising pulse, it is inverted by gate IC1a, which is also used as a buffer. Gate IC1b is used to invert a falling pulse a second time to put the pulse back into its original form. Note that pin 1 of gate IC1a is held high so that the input (pin 2) will only present one standard TTL load to the circuit under test (as compared to gate IC1b where the 2 inputs are tied together).

R1 is used as a pull up resistor to the pin 9 input to gate IC1d for noise immunity. Resistor R2 is used as a current limiting resistor for LED1.■

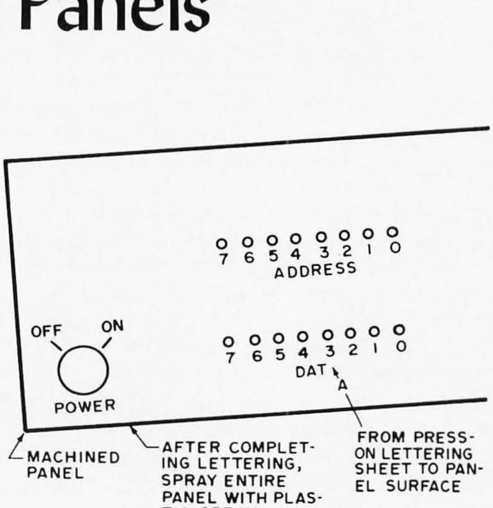
Dressing up Front Panels

Don R. Walters
3505 Edgewood Dr
Ann Arbor MI 48104

To dress up panels of equipment use press on lettering (available from stationery shops, college and university book stores, and from graphics arts supply shops) to label the various functions performed by the components which will be mounted on the panel (see figure 1). The lettering can be pressed on, wire brushed, chemically etched, or painted surfaces. The only caution is that the surface be free of dirt and grease before applying the lettering to the panel.

When applying the lettering, if a mistake is made the mistake can be removed by gently scraping the lettering off the panel.

After all the lettering is completed and you are satisfied with the job, lay the panel on a flat surface and carefully spray the panel with several (5 to 10) light coats of a clear plastic spray. Allow the surface to dry before applying the next coat. Also allow the sprayed surface to dry completely (overnight) before installing the components on the panel. The plastic spray protects the lettering from being rubbed or scraped off easily.



A word of caution; let the spray dry thoroughly before mounting the components. Secondly, on each spraying put down only a light coat. Otherwise the letters will tend to move (float?) a little out of place. Also be careful when mounting components onto the panel so that the lettering and/or the panel's surface is not scratched. The end result of lettering and care will be a very good looking piece of equipment.

Classified Ads Available for Individuals and Clubs

Readers who have equipment, software or other items to buy, sell or swap should send in a clearly typed or printed notice to that effect. The notices are free of charge and will be printed one time only on a space available basis. Insertions should be limited to no more than 100 words. Notices can be accepted from individuals or bona fide computer users' clubs only. Commercial advertisers should contact Virginia Peschke at BYTE for the latest rate card and terms.

SELL OR SWAP: Boards and main frame for Uniscope 100. Keyboards. Kleinschmitt Line Printer, TTY BRPE11 Punch, TTY Type 270 reader. Misc. 7400 series ICs. PDP-8e interface design for HSR. And as they say, much more. Roger Geffen, 71 Moore Road, Wayland, MA 01778, 617-358-2862.

TRADE OR SELL: IBM 360 to NOVA interface board, DEC input-output boards, dial up modems 202's various types. Can use components, W/W proto cards, TEK 453 or 454. Larry Skibicki, PO Box 47, Hollis NY 11423.

BUY OR SELL: Will sell one set of PC boards for SWTPC TV Typewriter I, or will buy one set of ICs and connectors for TVT-I. P.F. Curtis, 2146 Lost Tree Way, Bloomfield Hills MI 48013.

TVT 1 for sale, fully functional, half duplex operation 110 baud. Built in serial IO (RE Feb 75 pg 51) having cassette IO also. Also included telephone originate Modem 1070-1270 XMIT, 2025-2225 RCV. Two full pages of memory. (Southwest Tech) Keyboard mounted in attractive wooden box with power supply. Included all interconnecting cables and documentation. You only need a TV monitor. Asking $325 postpaid USA. Reason for sale: getting ASR-33 hard copy. Call or write C. Zalenski, PO Box 92, Johnson City NY 13790, 607-797-5777.

FOR SALE: Tally reader and punch, Model 420 and Model 424. Excellent condition, $150 for both. Lou Carbaugh PO Box 393, New Cumberland PA 17070.

FOR SALE: Teletype Corp 5-level 100 WPM (10 CPS) Typing Reperforator (Paper Tape Punch), 20 ma serial in, w/o motor, Model LBXD9, excel. condition $25. Teletype Corp 5-level 100 WPM (10 CPS) transmitter-distributor (PT reader) serial or parallel out, w/o motor, model LPR45, excel. cond., $25. Teletype Corp. 8-level punch model UPE800 for ASR33, like new, less than 20 hours use, $40. Friden Flexowriter automatic typewriter with 6-level PT reader and punch, upper and lower case, $100. A. Walker, 6000 N. 27th St, Arlington VA 22207.

WANTED: Back issues of BYTE. I subscribed late and did not get issues No. 1 to No. 3. Make an offer. R. I. Demrow, 11 Linda Road, Andover MA 01810.

TV Color Graphics

Don Lancaster
Synergetics

A graphic display lets us add pictures, lines, and symbols to the basic letters and numbers of an ordinary TV typewriter so that it can be used in chess, for video art, for games, printed circuit layouts, logic diagrams, charts, and so on. For many TVT and microcomputer uses, the ability to display several colors at once is very important; for others, it is simply a convenience that may not be worth the extra complexity.

Dots or Vectors?

There are two fundamentally different approaches to graphic displays. One is called the stroke method. The display is told where to start each symbol, which direction it should take, what color to display, and how far to go before stopping or changing direction. On the other hand, dot matrix method breaks the display down into as many raster scanned boxes or dots as a display refresh memory can provide. These blocks are lit or not lit with a selected color as needed to form an image.

The stroke method somehow sounds better, particularly if we are interested in curved figures and smooth edges on diagonal lines. But it has problems. Its most serious shortcoming is that it is not directly compatible with either a stock black and white or a stock color TV set. A special display is thus needed, although an oscilloscope can be substituted for single color, low resolution use. Another limitation is that the display and memory bandwidths have to be extremely wide, or else the display has to slow down its refresh rate as more and more symbols are added to the display. This can lead to unacceptable flicker even with a special tube phosphor.

The raggedness of the dot matrix system can be minimized by using enough memory to increase the resolution, and by minimizing the curved or diagonal lines to be displayed. The dot matrix technique has one major advantage in that we can easily and efficiently mix graphics and alphanumerics, simply by using a TVT system in a graphic mode on one field and in an alphanumeric mode on the second, merging the two separate displays as one. Besides giving more pleasing characters, this approach has an efficiency advantage – only six bits are needed to store a character, but 35 or more bits are needed to store and display the extended dot matrix character in a 5 by 7 format.

It turns out that a dot matrix display for graphics only is actually easier and simpler to build than an ordinary TVT; the character generator is no longer needed, and the high frequency timing is usually very much simplified. This type of display lends itself very well to existing microprocessor organization without extensive interface.

One limitation of the graphics dot matrix approach is that quite a bit of memory is needed, one bit corresponding to each and every possible dot location.

Some Television Limitations

What are the limits to the matrix size we can display on a television set? First and foremost, this depends on the amount of memory available because one bit will be needed for each dot position, and a fractional part of a bit (more on this later) will be needed for that location's color information. If we are also interested in *gray scale* (brightness variations), more than one bit per dot position will probably be needed. Gray scale techniques may not be worth the added cost and complexity.

Black and white sets can allow very dense

matrices. For instance, if we have a black and white set modified to display 72 characters in a 7 by 9 dot matrix with two vacant dot positions between characters, we end up with a horizontal resolution of 648 dots over the active scan. (This takes direct video and some modifications.) The scan rate limits our dots vertically. Without interlace, we get 262, 262.5 or 264 dot resolutions. If we have to, we can go to full interlace and double these values. Thus, it is quite possible to have a 512 by 512 dot graphic display. The memory for this display would take 262,144 bits, or 32,768 eight bit bytes. At 0.1 cent per bit, such a memory costs around $264. Besides being expensive,the density pushes the line pair resolution of the TV. Black and white sets can handle just about anything we can afford to throw at them in the way of graphics.

Color sets are much more restrictive, but still give us more than enough capability for very useful display matrices. An unmodified color TV has a maximum video bandwidth of 3 MHz brought about by the need to eliminate a 3.58 MHz subcarrier in the color processing video. Three MHz video is roughly equal to 6 million dots per second, which gives us around 380 or so dots per line. This doesn't include an allowance for horizontal retrace, so approximately 256 dots horizontally is a good limit. Similarly, if we don't use interlace, 256 dots vertically is also a reasonable limit. A 256 by 256 display contains 65,536 dot locations and costs us around $65 for a single color display at 0.1 cents per bit, and somewhat more if multiple colors are to be added.

Even if we were to separate totally the luminance and chrominance channels on the color TV (A major job that eliminates use of the TV for anything else), we'd still be limited by the number of holes in the shadow mask to resolutions only somewhat better than we can get on an unmodified color TV.

One very important thing to note is that the color bandwidth is far less than the dot resolution on a color TV set. In addition, the color bandwidth varies with the colors in use. The available bandwidth ranges from a high of 1.5 MHz for an orange cyan display to a low of 0.5 MHz for a red blue green tri-color display. Thus, with a high resolution display, it is not possible to change the color of each and every successive dot in the horizontal direction. For full color presentation, something less than 50 color changes across the screen is pretty much an upper limit, particularly if we are to minimize fringing and edge shading effects and are using saturated colors.

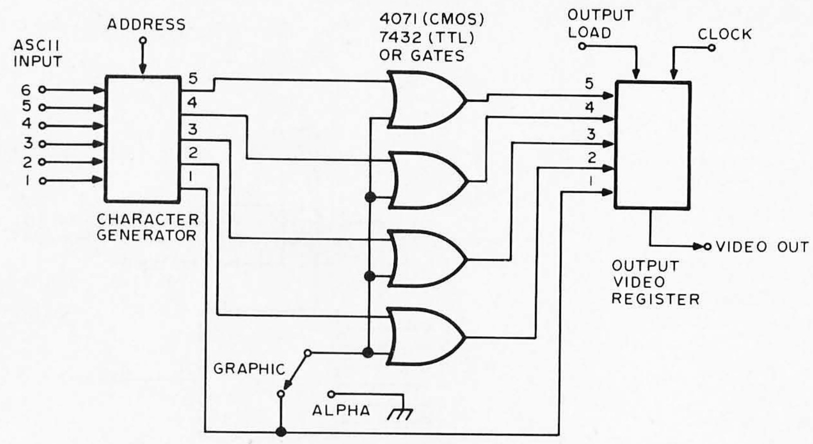


Figure 1: An ordinary alphanumeric television typewriter can be converted into a simple (but limited) graphics display by adding four OR gates as shown here.

The limited color bandwidth is both good and bad for graphics displays. The lower horizontal color resolution means that colors should remain constant for several dot positions at a time in the horizontal direction and whenever possible, we should change colors on a black dot or blanked portion of the display. Since this is the best we can do we use only a fraction of a bit per dot to specify the dot color. This means that our color storage won't add much to the basic memory size and cost. For instance, on an eight bit word, two color bits can specify any of four colors to six different dot locations. Or one color bit could specify a choice of two colors to seven different dot locations. Fortunately, black dominates and counts as wide bandwidth luminance video rather than color, so we can shift between black and colors and back again at full tilt.

Some Simple Graphics Displays

An ordinary alphanumeric TVT can be converted into a very limited performance graphics display simply by adding one quadruple OR gate package as shown in figure 1. The OR gate goes between the character generator ROM outputs and the output video shift register. It monitors the leftmost dot output and makes the other dots 1 if the leftmost dot is a 1. If you type a space, you get a blank location. If you type a D for a dot, you get a white box at each and every character location specified. There's nothing magic about using a D; any character with all leftmost dots which are a 1 will do, such as B, E, F, H, K, L, M, etc. . .

The resolution of your display depends on the original format, for you get one box

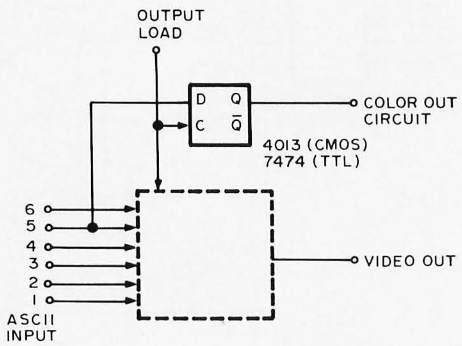


Figure 2: By using one of the six input lines to control the state of a flip flop, a two color display can be achieved. Use the color out line to drive the color modulator.

per character. With a 32 character per line display of 16 lines, you get a total of 512 "black surround" boxes in your matrix. While this isn't very attractive or much in the way of resolution (What do you want for only 14¢ extra in circuitry?), it is handy for studying graphics techniques, and is of limited use in games, displays, simple video art applications, and for generating large alphanumeric characters.

Figure 2 shows how to add color. Since we used only one bit of the word to store the alphanumeric information, we have enough bits left to handle quite a few different colors, ranging from 32 colors on a six bit memory word to 128 on an eight bit memory word. The color information is removed directly from the character storage memory, monitoring bit 5. The color output is delayed and resynchronized to the output video with the flip flop as shown. This delays the color to match the delay involved in the character generator. The color flip flop's output goes to a color subcarrier generator. The subcarrier is programmed to produce a red output when the fifth bit is a 1, and a blue output when it is a 0. It returns to reference phase during horizontal retrace.

Now, for no box, we type a space. For a red box, we type a R and for a blue box, we type a B. The first bit of the character generator *output* decides box or no box (luminance), while the delayed fifth bit of the character generator *input* decides red or blue.

We can pull a sneaky trick to double the resolution in the vertical direction, by making the top half of any box one color and the bottom half a second color. Better yet, we blank out enough of the middle of the box to exactly match what's blanked out *between* vertical boxes, so we get even spacing of all the boxes in the vertical direction. For instance, on a 5 by 7 dot matrix with three lines between vertical characters, we use only the two upper dots and the two lower dots, skipping the three middle ones.

Figure 3 shows the details. We carefully search the dot patterns of all the displayed characters to come up with some useful combinations. The inverse of bit one output is what counts in the double mode. By typing a B we get neither box in a character location. Typing an A gets us an upper box but not a lower box. A V gets us a lower box but not an upper one. And a < gives us both upper and lower boxes. Color may be added using the circuit of figure 2 connected to the seventh and eight memory bits.

A 96 by 96 Full Color Display

Let's look at the design problems involved in building a full color display that uses a modest memory of 2048 by 8 bits for storage (costing $16.34, at 0.1 cents a bit) and yet still has enough useful resolution for chess displays, Pong style games, video synthesis, creative art, and so on.

The central problem in designing a graphic display is proper *partitioning* of the memory. Partitioning simply decides what portion of the memory is related to the display in what manner. The three very important partitioning questions are:

1: How many resolvable elements in what positions and colors are supplied by a single memory word?

2: How is graphic information separated from alphanumeric information such as scores and printouts?

3: How do we separate the "seldom changing" memory portions of a display (such as the chess grid or a foul line) from the "often changing" parts (such as a just captured rook or a fast moving hockey puck) to simplify as much as possible the software and external control of the display?

For a 96 by 96 color display useful for chess and Pong style games, we might partition each memory byte as shown in figure 4. The first six bits of our eight bit

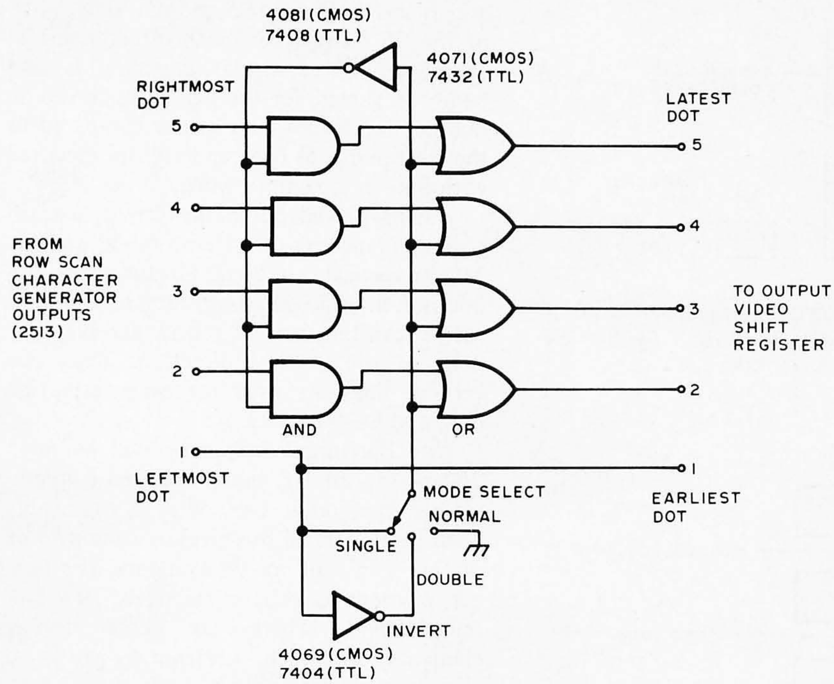


Figure 3: Logic required to generate two possible boxes within one character position of a TV Typewriter.

word specify the luminance (black or not black) of six dot locations; the final two bits specify the color of all six of those locations simultaneously. This gives us up to four colors plus black, and lowers the color change resolution enough that the color TV stays happy. In a Pong style game, the goal and sidelines can be one color and the opposing teams two additional colors, and the ball or marker yet another color. For chess, we can use a two-color chessboard with two-color opponents.

While our first six bits could be arranged as six successive dots horizontally or vertically, a grouping of three horizontally by two vertically seems to work out well for

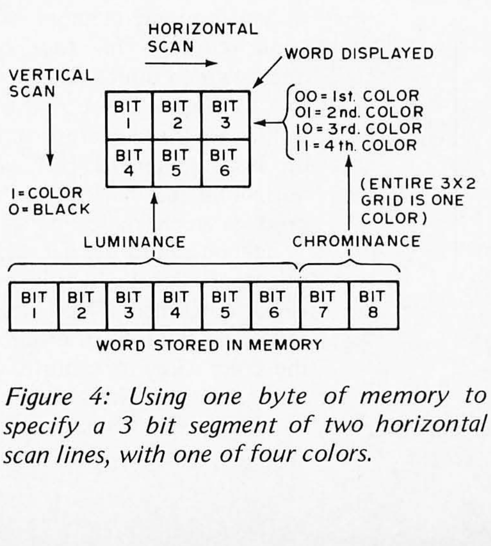


Figure 4: Using one byte of memory to specify a 3 bit segment of two horizontal scan lines, with one of four colors.

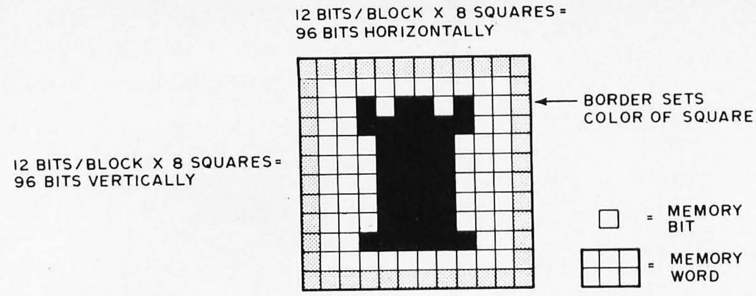


Figure 5: Setting a chess piece into a grid of colored squares of a color graphics display. For the chess board application, a grid of 8 by 8 = 64 squares is required.

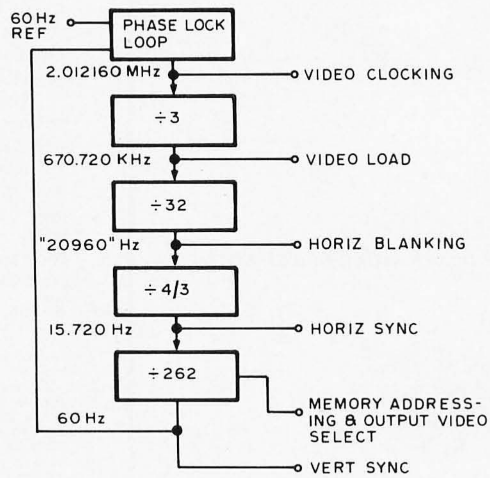


Figure 6: System timing for a 96 by 96 display grid. In this scheme, the phase lock loop is used to maintain synchronization with the 60 cycle line frequency.

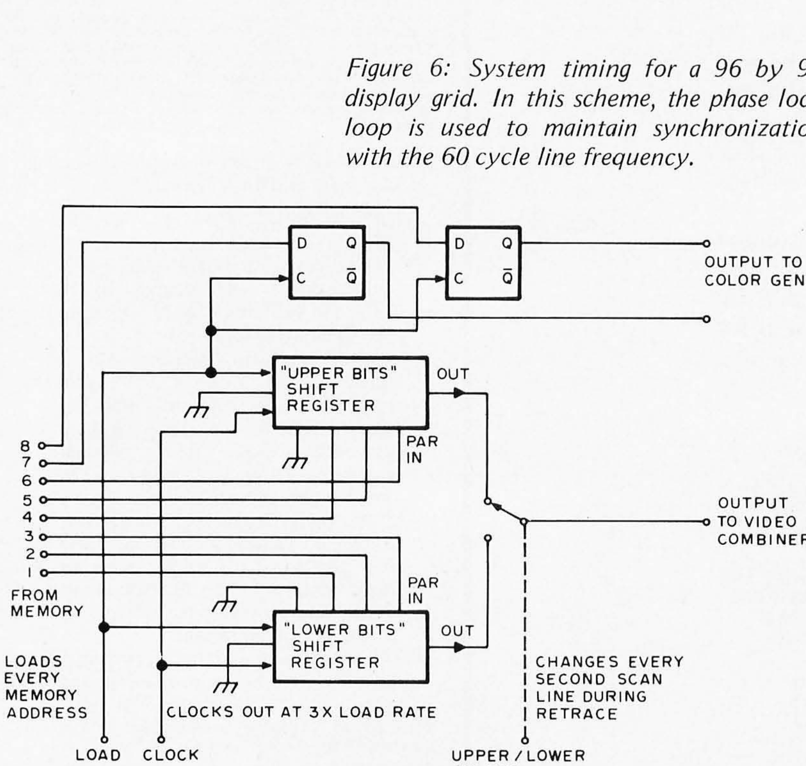


Figure 7: The video output circuitry for a graphic display is simpler than that of a character display. This logic implements the 3 by 2 format of figure 4.

chess and many other games. The basic chess grid is shown in figure 5 and uses one eighth of the 96 elements in both directions. The choice of a rectangular grouping is also handy in games, for the ball or puck can be double size in both directions compared to the goal and foul lines and still be specified with only one memory word.

Having picked our basic format, we can go on and design our system timing. We start with a vertical rate of 60 Hz and use a 262 line non-interlaced system, for a total of 192 active scan lines and 70 retrace and blanking lines in the vertical direction. Thus two vertical lines are used for each resolution box, and 96 * 2 = 192.

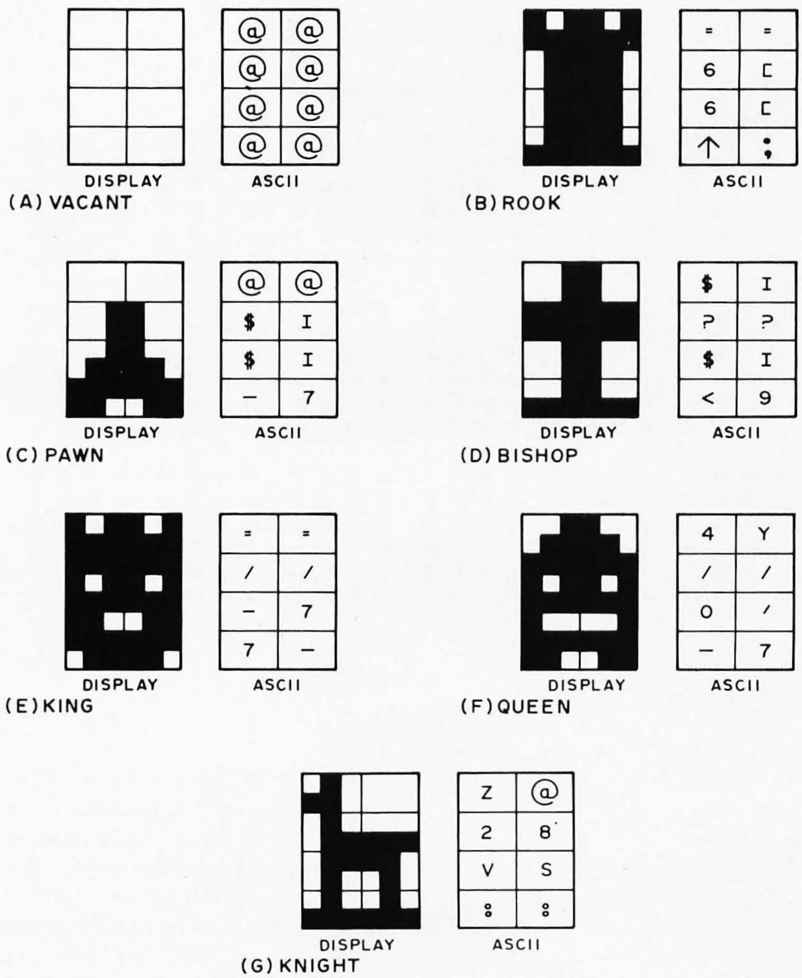
Our horizontal frequency will be 60 * 262 = 15,720 Hz, equal to a 63.6 microsecond horizontal line. With a reasonable horizontal retrace, this gives us around 48 μs of live scan time for 96 elements. The time per element is thus a relatively lazy half microsecond. There are three display elements for each horizontal word, so memory clocking will be needed at 1.5 μs intervals. One possible system timing setup is shown in figure 6.

The output video circuitry is simpler than in an alphanumeric display since we don't need a character generator. Figure 7 shows one possible arrangement. The memory words are clocked once every 1.5 μs and three bits are loaded into a four bit shift register and clocked out at a 2.01216 MHz rate. The other three memory bits are loaded into a second identical shift register. At the same time, the two color bits are decoded and sent to the color subcarrier system.

The registers are selected on alternate scan lines. This automatically picks the first three bits for the upper three boxes and the second three for the lower three boxes of each memory. Note that we can use a data selector since we only change it during the horizontal retrace time. The whole job can not be done with selectors due to the glitch and settling time problems. Note also that we get the same memory words back in the same sequence for *four* consecutive scan lines, two to display the first three bits and two to display the second three bits. This is similar to the readdressing that takes place on an alphanumeric TVT when we get each horizontal line back at least seven times to produce an entire dot matrix character.

The output of the data selectors is a video brightness signal which is routed to a typical video combiner. The color information is routed to the subcarrier circuit. Be sure that the color circuitry returns to the reference phase during the horizontal retrace time. Try to design the display so that most color

Figure 8: A set of chess men constructed using the video output circuitry of figure 7: The diagram shows a 6 bit ASCII subset character pattern to the right of each figure to show what should be loaded into the memory.



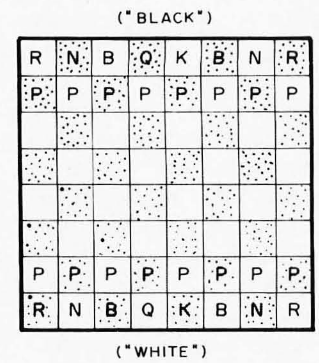


Figure 9: The starting configuration of a chess game is established by repeating the appropriate character patterns at selected places in the display memory and surrounding the piece patterns by the **border** *pattern. If the "o" character is used to represent the 8 elements of the chess man selected from figure 8, then the pattern for each piece is:*

O G G '
C o o $
C o o $
C o o $
C o o $
C o o $
9 8 8 <

Each "o" should be replaced by a character from one of the chess patterns of figure 8. Since the ***border pattern is fixed,*** *moving a chess piece is accomplished by moving only the 8 characters in the center of each board position. Software for a chess display would ini-* ***tialize the border pattern*** *at the start of play.*

changes take place on a black or blanked location. If color fringing becomes a problem, a slight additional amount of video delay (with respect to the color changes) may minimize this effect.

Using this circuitry, each chess piece requires 8 characters of memory. Figure 8 shows one possible arrangement of the chessmen for a chess display (along with 6 bit ASCII subset character codes). The program of figure 9 shows us the starting position for a chess game. A basic hockey or Pong setup is shown in figure 10 with its program.

We can add scoring or other alphanumerics most simply by using an ordinary TVT with the graphics display and alternating frames, as shown in figure 11A, or by summing video as shown in figure 11B. Both system timing circuits must be locked together and have the same number of horizontal lines, and the same amount of horizontal retrace must be provided in both modes if the television is to recognize a continuous program. Another possibility is to use simple box "stroke style" characters and store them directly in the memory matrix.

Figure 12 shows several approaches to the third partitioning problem, the separation of often and seldom changing data. Most often, a graphics TVT display will be used in association with a microprocessor to get extensive game capabilities. This has a tremendous advantage over wired logic in that we can change rapidly a game format simply by changing programs.

The object of memory partitioning is to trade off specialized timing against the total number of memory bits that have to be involved with every computation and up-

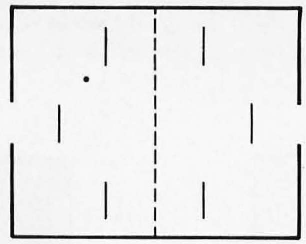


Figure 10: Display pattern for a Pong style video game. This figure shows the contents of display refresh memory for the upper half of a playing field display. The lower half is a mirror image except for the location of the ball (which would be controlled by the software of your microcomputer.) The characters shown are 6-bit ASCII subset characters which give the proper codes for the 3 by 2 format of the video graphic generator of figures 4 and 7.

```
O G G G   G G G G   G G G G   G G G V   G G G G   G G G G   G G G G   G G O @
I @ @ @   @ @ @ @   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   @ @ @ J   @ @ @ J   @ @ @ J   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   @ @ @ J   @ @ @ J   @ @ @ J   @ @ @ @   @ @ @ @   @ @ I @

I @ @ @   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   @ @ @ J   @ @ @ J   @ @ @ J   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   [ @ @ @   @ @ @ J   @ @ @ @   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ I @

I @ @ @   @ @ @ @   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ @   @ @ @ @   @ @ I @
I @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ I @
@ @ @ @   J @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ @ @   @ @ J @   @ @ @ @
@ @ @ @   J @ @ @   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ @   @ @ J @   @ @ @ @
@ @ @ @   J @ @ @   @ @ @ @   @ @ @ J   @ @ @ @   @ @ @ @   @ @ J @   @ @ @ @
```

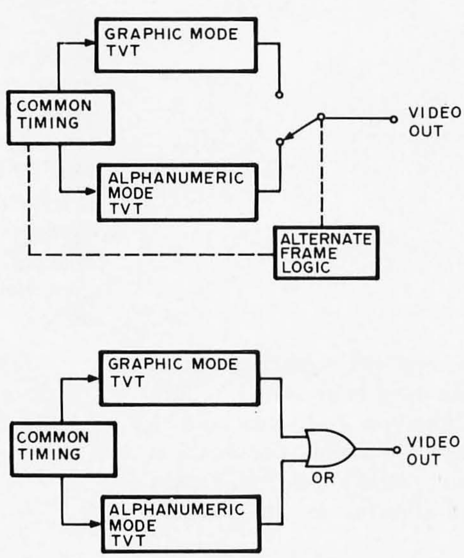


Figure 11: Combining a graphic display generator with a character display generator for a single television set. One way is to alternate on every other frame using a digital switch. A second method is to sum the video signal (using the OR gate) producing a combination output.

date. For instance, in figure 12A, we simply use the entire 2 K by 8 bit memory in one brute force piece. While conceptually the simplest and most flexible, it takes extensive software and a long time to fill the display initially and then to specify any changes.

In figure 12B, we have added some dedicated timing that separates the chess playing spaces from the board spaces. This is done with some "crosshatch" logic gates that pick out the places the chessmen are to go. Now a much smaller portion of memory is involved in specifying piece locations. While figure 12A requires 1536 words, the design of figure 12B necessitates 512 words.

Can we do better? In figure 12C, we've broken our memory into three distinct areas and selected them with wired logic. The main area is the board background which can be a ROM or a RAM which is loaded once and never changed. The second area is a file where we have stored the programs for the six chessmen involved (pawn, bishop, knight, rook, king, and queen). This file can repeatedly be addressed to go in different locations in the main graphics display.

Finally, we have a small active memory that decides what chess piece of what color (or the option of no piece) goes in what location. This active portion of our memory can be as little as 32 words since there are only 64 squares and only four bits are

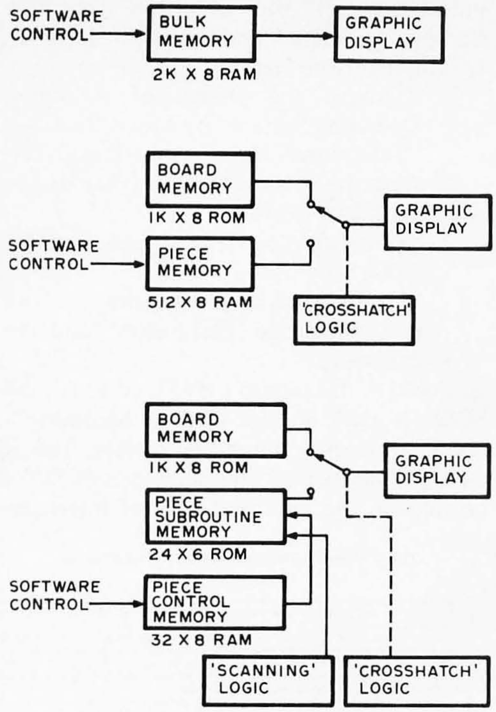


Figure 12: Three ways of organizing the refresh logic. At A, the most general approach is taken, with a full refresh memory for the entire screen. In this approach, computer software completely controls placement of patterns. At B, the memory is reduced slightly by partitioning it into two segments: One segment has the fixed "board layout" which is set up at the start of a game (or burned into the ROM of a commerial product); the second segment has the varying information on placement of the game pieces, and a crosshatch switch electronically combines them. Finally at C, even further specialization is introduced with three segments. The board memory remains, as in option B. Each piece of a chess grid is presented as a special ROM which is fixed with the bit patterns needed to generate any one of the 6 possible chess men. The placement and color are specified by a final RAM segment which changes during the game.

needed to specify any chessman or blank of either color. Here we trade the simple generality of a 2 K by 8 refresh memory off against the specific optimization of the 3-tiered memory for board games. Only a small amount of dedicated logic is needed for this specialization.

In a Pong style game, the ball is specified as four bits of the six bit word, making it twice the size of the background, foul, and goal lines. Ball motion can usually be computed during the vertical retrace time. For instance, if our microprocessor has a 10 microsecond cycle time, and we have around 4 milliseconds of vertical retrace time, up to 400 machine cycles are available to compute the new ball location.

The 96 by 96 format seems to lend itself well to game graphics and seems easy and reasonable to build. It is by no means the only possible organization for graphic displays. The basic partitioning problems will change for different types of graphic displays.■

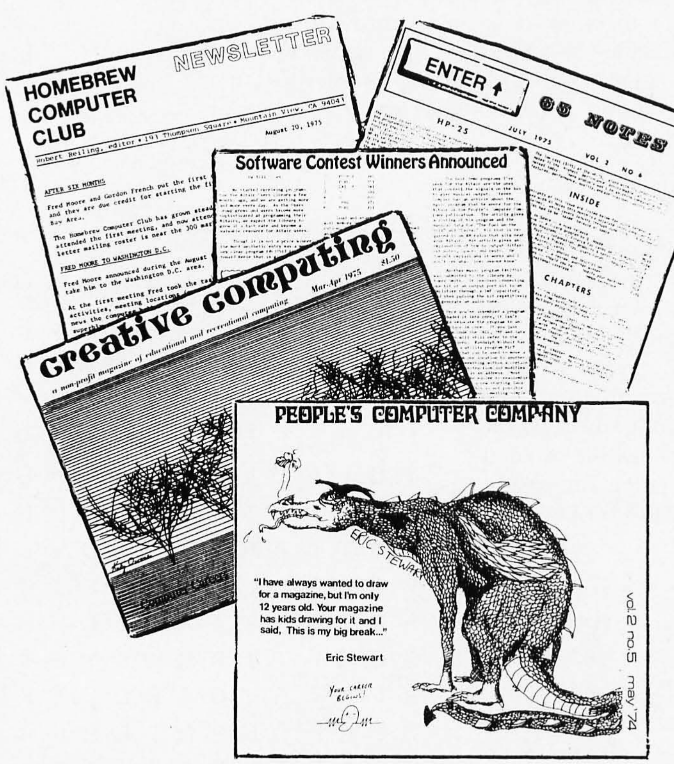
NUMBERS

Robert Baker
34 White Pine Drive
Littleton MA 01460

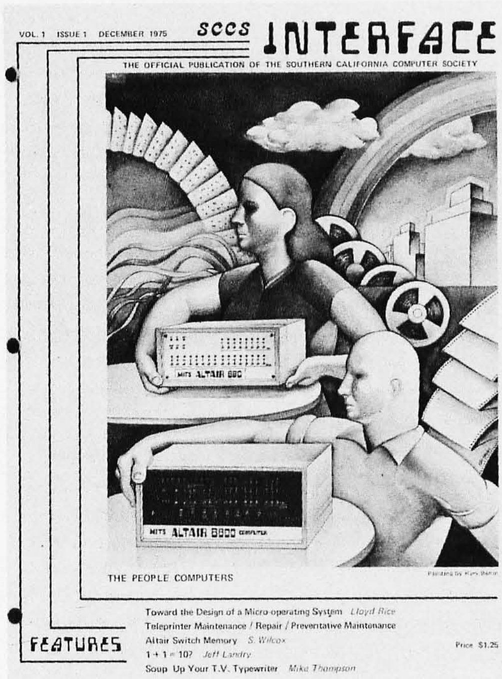
Hidden in the matrix are 39 words related to numbers. Find a word, circle it or color it in, and cross it off the list. Words may be forward, backwards, up, down, or diagonally but always in a straight line; never skipping letters. However, some letters are used more than once. After circling all the words on the list, the unused letters will spell the name of one of the holidays this month. The answer key will follow in next month's BYTE.

BILLION
BINARY
COMPLEX
DECIMAL
EIGHT
ELEVEN
EXPONENT
FIFTEEN
FIFTY
FIVE
FORTY
FOUR
FRACTION
HUNDRED
IMAGINARY
INTEGER
LOGARITHM
MILLION
MODULUS
NEGATIVE
NINE
ONE
POSITIVE
RADIX
RATIONAL
REAL
ROOT
SEVEN
SIX
TEN
THIRTEEN
THIRTY
THOUSAND
THREE
TRILLION
TWELVE
TWENTY
TWO
ZERO

G	M	N	E	G	A	T	I	V	E	L	E	V	E	N
X	I	D	A	R	P	O	S	I	T	I	V	E	R	D
N	L	Y	T	A	M	H	T	I	R	A	G	O	L	E
O	L	T	R	T	N	O	Y	T	N	E	W	T	I	R
I	I	F	I	I	E	O	D	E	V	I	F	M	D	D
T	O	I	L	O	E	C	W	U	Y	O	A	U	N	N
C	N	F	L	N	T	N	O	T	L	G	D	E	A	U
A	T	T	I	A	R	N	R	M	I	U	V	I	S	H
R	E	E	O	L	I	I	O	N	P	L	S	G	U	L
F	N	E	N	N	H	T	A	I	E	L	H	H	O	A
S	O	N	I	T	T	R	H	W	L	Y	E	T	H	E
B	I	N	A	R	Y	O	T	R	O	L	T	X	T	R
G	E	X	P	O	N	E	N	T	E	R	I	R	O	U
D	A	D	E	C	I	M	A	L	Y	E	E	B	O	O
N	E	V	E	S	I	N	T	E	G	E	R	Z	R	F



Clubs and Newsletters



An Excellent Publication

BYTE congratulates the Southern California Computer Society on the progress of its publication, **Interface**, from an excellent newsletter into an even better magazine style publication with 32 interior pages plus cover. SCCS **Interface** is "the official publication of the Southern California Computer Society," and is published monthly. The first issue features:

"Toward the Design of A Micro-Operating System" by Lloyd Rice
"Teleprinter Maintenance/Repair/Preventative Maintenance" (Part I) by Cliff Sparks
"Soup Up Your T.V. Typewriter" by Mike Thompson
"1 + 1 = 10?" by Jeff Lendry
plus numerous "featurettes" and departments.

As noted in this month's BYTE editorial, the SCCS is well on the way to becoming a national personal computing society. This is clearly indicated by the objectives of SCCS printed on the masthead page of **Interface**:

THE SOUTHERN CALIFORNIA COMPUTER SOCIETY

OBJECTIVES

A non-profit organization whose purpose is:

A. To exchange and disseminate information among the Society members concerning the computer arts and sciences.

B. To provide a technical assistance to other members of the Society in those computer projects which are not undertaken for pecuniary gain or profit including but not limited to hardware, software, and computer programming.

C. To publish books, newsletters, magazines, and other periodicals for the benefit and education of the Society members and the general public.

D. To conduct and sponsor seminars, lectures, and courses relating to the computer arts and sciences.

F. To develop and maintain computer centers and laboratory workshops for the members of the Society and the general public including provisions for time-sharing operations.

To receive **Interface,** membership is required. This can be obtained by sending $10 along with your name, address, business phone and home phone to:

Southern California Computer Society
PO Box 987
South Pasadena CA 91030

or call 213-682-3108 and leave a message.

SCCS Branches Way Out – to the Delaware Valley

The Southern California Computer Society is branching out. Word of the Delaware Valley chapter of the society reached BYTE late in November. The contact person for this chapter is:

Martin Dimmerman
1228 Barrowdale
Rydal PA 19046

The Delaware Valley chapter will be of interest to persons in the Philadelphia PA and Camden NJ area. Martin's phone number is 215-887-6992.

Attention HP-65 Hardware Hackers

The October-November double issue of the HP-65 Users Club *65 Notes* came BYTE's way recently. This issue is of special significance to those individuals who want to modify and add on to an HP-65 machine. "Inside the HP-65" is a special 14-photo series of detailed pictures of the innards of

the HP-65 with a reference grid on each picture – provided by William Kolb who dissected his own machine for the purpose. The photos are accompanied by complete notes on the dissassembly process. Other hints in the same issue include how to make HP-65 cards from other readily available magnetic media, how to split the enter key to obtain extra functions, information on an HP-65 printer interface (including photos) designed by Heinrich Schnepf of West Germany, and a write protect defeat switch for the tape drive.

The *65 Notes* is published by Richard J. Nelson, 2541 W Camden Place, Santa Ana CA 92704, and regularly features extensive information on the HP-65 and other programmable calculators.

All It Takes Is a Little CACHE

Bill Precht sent BYTE a note just as the February issue goes to press, enclosing word of the evolution of the Chicago area activities. The organization is now known as CACHE (Chicago Area Computer Hobbyists' Exchange), and a mailing address has been established:

CACHE
PO Box 36
Vernon Hills IL 60061

Bill sent along the latest issue of the CACHE newsletter along with word of the name change.

Micro Fun in the Big Apple

Robert Schwartz sent BYTE a short note that the New York City Micro Hobbyist Group had its second meeting Friday December 12 at LaGuardia Community College in Long Island City, Queens. Interested parties in New York City can contact Robert by mail at

375 Riverside Drive, 1E
New York NY 10025

or by phone at

212-663-5549 (evenings).

Report:

BYTE's Audio Cassette Standards Symposium

Manfred and Virginia Peschke

BYTE Magazine sponsored a symposium on November 7 and 8, 1975 in Kansas City MO regarding the interchange of data on inexpensive consumer quality audio cassette drives. These drives may be used as one of the mass storage devices in the first generation of personal computers, and will retain importance for some time to come as a means of interchange of software between computer enthusiasts who purchase products of the small systems industry. In order to promote the growth of the industry, BYTE sought to achieve an industry standard on audio cassette data interchange through a working conference.

We extend our greatest appreciation to the 18 people who worked very hard until late Friday night and Saturday morning to discuss the multitude of problems and solutions associated with digital recording on auto cassettes. The names of the participants are listed in table 1. In spite of the short time available, the participants were able to draft a set of provisional standards which seems to promise great reliability and is rather inexpensive to implement; implementations may be entirely in hardware, or may require a mix of software and some minimal hardware. Considerations were given to the problems of speed variation among recorders and playback equipment, start and stop delays, recording density (or speed) versus reliability, and recording frequencies to avoid interference with the telephone network in case some users plan to transmit the tones of the cassette over the phone lines.

On Saturday afternoon, Mr. Felsenstein and Mr. Mauch volunteered to write up the consensus among the participants as to a provisional standard which has been reproduced below.

Provisional Audio Cassette Data Interchange Standard

The consensus among the participants of the audio cassette standards symposium at Kansas City MO sponsored by BYTE Magazine is as follows:

The proposed standard centers around the use of a frequency shift modulation method from which serial clock data can be extracted at rates of up to 300 baud. The system is intended to be used with low to medium cost cassette recorders incorporating electrical stop and start capability which may be operated under program control.

The technique proposed provides for long and short term tape speed variation, limitations in bandwidth due to effects such as tape misalignment, and the necessity to retain low cost and low complexity of the hardware. The technique allows for potential operation at higher tape speed than the nominal 1.875 inch/s (4.75 cm/s).

A mark (logical one) bit consists of eight cycles at a frequency of 2400 Hz.

A space (logical zero) bit consists of four cycles at a frequency of 1200 Hz.

A recorded character consists of a space as a start bit, eight data bits, and two or more marks as stop bits.

The interval between characters consists of an unspecified amount of time at the mark frequency. In this respect the data format is similar to that of asynchronous data communication.

The eight data bits are organized least significant bit first, most significant bit last, and followed (optionally) by a parity bit. The total number of significant bits and the parity bit cannot exceed 8.

Where less than eight data bits are used, the unused bits (following the optional parity bit) at the end of the character are mark bits (2400 Hz).

Data will be organized in blocks of arbitrary and optionally variable length, preceded by a minimum of five seconds of marks.

To avoid errors due to splice and wrinkle problems common at the begining of tape, the beginning of the first data block will occur no sooner than 30 seconds from the beginning of clear leader.

The contents of the data block are not specified.

The data block ends after the stop bits of the final character.

Bit clocking information may be extracted from the recorded waveform, which is always an integer multiple of the bit rate, regardless of tape speed. This permits the recovery and retiming of data by means of a UART, which requires a clock of sixteen times the bit rate, although other simple circuitry may be used.

A reliable bandwidth of 3000 Hz was assumed in choosing mark and space frequencies due to the head misalignment expected between various cassette recorders. The recording technique is a redundant form of Manchester or bifrequency code which has a long history of reliability in the computer industry. In its present form it was proposed by three independent manufacturers at the conference. One cited reliability rates of one error in 10**7 characters for 200 passes. ■

Table 1: Participants at Audio Cassette Symposium.

Ray Borrill, 1218 Prairie Dr, Bloomington IN

Hal Chamberlin, The Computer Hobbyist, P O Box 5985, Raleigh NC 27607

Tom Durston, MITS, 6328 Linn NE, Albuquerque NM

Lee Felsenstein, LGC Engineering, 1807 Delaware St, Berkeley CA 94703

Joe Frappier, Mikra-D, 32 Maple St, Bellingham MA

Bill Gates, MITS

Gary Kay, Southwest Technical Products Corp, 219 W Rhapsody, San Antonio TX 78216

Bob Marsh, Processor Technology, 2465 Fourth St, Berkeley CA 94710

Harold A Mauch, Pronetics, 4021 Windsor, Garland TX 75042

Bob Nelson, PCM, San Ramon CA

George Perrine, HAL Communications Corp, Box 365, Urbana IL 61801

Ed Roberts, MITS

Richard Smith, The Computer Hobbyist, P O Box 5882, Raleigh NC 27607

Les Solomon, Popular Electronics, 1 Park Av, New York NY 10016

Michael Stolowitz, Godbout Electronics, Box 2355, Oakland Airport CA 94614

Paul Tucker, HAL Communications Corp

Mike Wise, Sphere, 791 S 500 W, Bountiful UT 84010

Bob Zaller, MITS

A View From the Silicon Valley

Jim C Warren Jr
Star Route Box 111
Redwood City CA 94062

Silicon Valley, better known as the Santa Clara Valley (or Silicon Gulch by the more vitriolic) is located about forty miles south of San Francisco. It is the undisputed capital of the semiconductor industry in general, and microprocessor manufacture in particular, with other major outposts being located in Massachusetts and Texas. It serves as headquarters for American Micro Systems, Fairchild, Hewlett-Packard, Rolm, Intel, Intersil, Teledyne, Varian, Anderson-Jacobson, Signetics, and Four-Phase . . . just to name a few. Expanding slightly to cover the San Francisco Bay Area, we can pick up a multitude of other companies of interest to the computer hobbyist, including Godbout Electronics, and Diablo. To put it mildly, *much* is happening out here. Here are a few about-to-be's:

CallComputer, a super cheap time-sharing service well-known to and well used by the Homebrew Computer Club crowd, has recently installed a direct connection to the Los Angeles area. It is actively supporting the needs of computer hobbyists, soliciting their business, and offering significant assistance. It has contacted the Southern California Computer Society hobbyist group, offering its services to them. Among other things, one service will be to allow Bay Area hobbyists to converse with Los Angeles hobbyists, toll-free, via the computer link. This communications line can support up to six simultaneous 30 CPS (characters per second) bidirectional links, and more links at lower CPS. Their minimum connect time charge is 99 cents per hour (midnight to 6 AM). They offer Hewlett-Packard BASIC, and Data General's ECLIPSE software systems, including BASIC, FORTRAN, ALGOL and assembler languages. (Call Computer, 1961 Old Middlefield Rd, Mountain View CA 94043).

Godbout Electronics, a regular BYTE advertiser, plans announcing a new computer kit for hobbyists built about MSI (medium-scale integrated circuits). It executes the PDP-11 instruction-set and is quite low cost (Godbout Electronics, Box 2355, Oakland Airport CA 94614).

Computer Converser, Inc, a spin-off from Call Computer, plans to announce a TV based terminal for only $595. It is assembled and ready to use, includes a good-quality acoustic coupler, is switch-selectable for 10 or 30 char/sec and full or half duplex operation. It operates with any home TV, displaying twenty six 40 character lines, and can replace a TTY (Teletype Corp. ASR33). In the latter part of January, they expect to begin marketing the acoustic coupler portion of this terminal as a separate item for $95. They are also planning a June announcement of a complete system that will include the above terminal and coupler, plus a microprocessor, plus an 80 character line video monitor, plus BASIC software for around $1200, assembled. Incidentally, the terminal is about the size and weight of a small typewriter, thus easily portable. All of this gear is offered with a full warranty; they are a new and small company and most eager to gain a reputation for good quality and backing their products. (Computer Converser, Inc, 1961 Old Middlefield Rd, Mountain View CA 94043).

New-goodies rumor-mongering is a favorite activity among Silicon Valley engineers and technicians at professional meetings or martini brunches. One rampant rumor is that several chip houses (semiconductor manufacturers) are on the verge of announcing the digital to video chips. Though the details are vague, the general idea is that these chips will be capable of, for instance, accepting binary coded brightness information and horizontal and vertical coordinates for a sequence of points. The chip will then buffer this information, and use it to generate video signal output acceptable to a standard TV. Another rumor is that Intel is working on a simplified controller for their 3000 series, bit sliced microprocessor. The 3000 series is

a sophisticated and powerful chip set, however, partially due to the complexity of the current controller chip organization, it is definitely not for the computer amateur.

Nybble is a term rapidly gaining popularity among digital semiconductor pros, around here. Obviously, a nybble is part of a byte. Back in the early 1960's, IBM introduced byte as the term to specify a portion of a 32 bit word; namely an 8 bit portion. Other manufacturers picked it up, some even using it to refer to a 6 bit portion. Now, with 2 bit and 4 bit bit-sliced microprocessors being developed by the dozens, nybble is becoming popular for referring to those 2 and 4 bit slices.

Who says this is the kook capital of the world? Eccentrics, maybe: Lee Felsenstein, furnishing great assistance and expert advice to Homebrew Computer Club members, heads up LGC Engineering in Berkeley. LGC stands for Loving Grace Cybernetics, of course. Bill Godbout, president of the company with the two computer kits and flashy ads in BYTE, has a PhD in Operations Research. George Morrow, an Associate with Godbout Electronics and project director for their MSI-11 design, is completing his doctorate in mathematics at the University of California in Berkeley. We may have some unusual ways, out here, but we sure turn out some dandy toys. ■

Could a Computer Take Over?

Ed Rush
PO Box 14369
Santa Barbara CA 93107

Just how ridiculous IS the idea of a computer deciding to take over the world and be its dictator?

Upon hearing this question, most people who are not computer oriented will laugh and say "That's only in science fiction stories." They will be much more likely to complain about "becoming a number," with everyone from the grocery store to the government wanting their number instead of their name.

Those who are more familiar with computers will laugh off the concept and charge it to paranoia due to ignorance. "A computer is little more than a lot of wires conducting currents here and there," they will say. "Besides, if it gets uppity you can always pull the plug."

However, that group of people who are both computer knowledgeable and fans of the art form known as science fiction, but more properly called speculative fiction, might ask "Can you always pull the plug? Could a computer really seize the reins of government? And if so, how?"

In trying to answer these last questions, let us first speculate on the capabilities the computer itself would have to have.

Super Computer

First, the computer system would have to be extremely powerful (in today's frame of reference). Considering the fact that computer technology is already far outstripping man's capability of harnessing it, a super computer is not hard to imagine in the not so distant future; perhaps even today in some secret government project.

While something on the order of 1000 computer circuits can now be stored in a cubic inch, only one such circuit would fit that space in 1960 and it took 20 cubic inches to hold one in 1950. A given number of programming instructions cost 1000 times as much in 1955 as in 1970, and probably 10,000 times today's cost, despite inflation. High speed computers now operate several thousand times as fast as they did in the early 1950s. Data storage capabilities are growing even faster. The capacity of an early 1970s system was a couple of million times that of 1955, and that is for a common large installation, not the maximum possible. The on line storage cost also shows a millionfold improvement since 1950 (Martin, James and Norman, Adrian R.D., *The Computerized Society,* pp. 9-14). Who is to say what 1980 will bring?

The next requirement is that this machine must be able to interact with changes in input from a multitude of input sources at once, a situation common to today's time sharing practices.

Such a machine must embody what is commonly called "artificial intelligence." That phrase is used hesitantly; since things which immediately provoke the description "artificial" are actually just natural materials rearranged by man. Intelligence is defined as:

> The capacity for knowledge and understanding, especially as applied to the handling of novel situations; the power of handling a novel situation successfully by adjusting one's behavior to the total situation; the ability to

"The first man to use a machine was the first of our primitive ancestors who picked up a rock to hurl at some passing animal or to crack open some edible nut. In the million-plus years since then, our machines have grown much more complex, but even in our modern era of computers, . . . their basic purpose remains the same: to serve man.

"Whether our machines truly serve us is a question much debated by science-fiction writers and other professional speculative philosophers. Does some essential quality go out of human life when it becomes too easy? Have our automobiles, telephones, typewriters and elevators sapped our vigor? Are we speeding into flabby decay because we have made things too easy for ourselves?

"And as our machines grow more able, when do they cross the boundary that separates the living from the unliving? Is it possible that we are building machines that will make humanity obsolete? Perhaps the day is coming when we ourselves will be rendered unnecessary, and our sleek successors, creatures of metal and plastic, will inherit the earth.

". . . Many a bitter attack on the encroachments of the machine age has been produced by a writer using an electric typewriter in an air-conditioned room, innocently unaware of the inner contradictions involved. We need our machines, but we fear them. . . ." Robert Silverberg, Introduction to *Men and Machines.*

> apprehend the interrelationships of presented facts in such a way as to guide action towards a desired goal. Psychologists still debate whether intelligence is a unitary characteristic of the individual or a sum of his abilities to deal with various types of situation. *(Webster's New International Dictionary of the English Language*, Second Edition, Unabridged, p. 1291.)

A machine with this capability would be an intelligence in its own right, not just an electronic mimic. It might take the form of a massive, immovable complex, or it might someday take form as a troop of man sized robots, or it might be a combination of these, with the latter as mobile extensions of the former.

Although Isaac Asimov has written extensively about the possibilities of robotics, most authors who have seriously considered a computer takeover have postulated the immobile complex. There are at least two good reasons for this assumption: First, such a machine would most likely be the first to have massive capabilities, and as such would most likely be far too big to move about. Second, it would undoubtedly require very heavy security as the most advanced piece of computer hardware in existence; protection not only from spies, but from vandals, intentional or otherwise. Examples of postulated massive complexes are HARLIE (Gerrold, David, *When HARLIE Was One*), Project 79 (Caidin, Martin, *The God Machine)* and Colossus (Jones, D.F., *Colossus: The Forbin Project).* The last two are built inside man made caves in the Rocky Mountains as the U.S. Air Force's North American Aerospace Defense Command (NORAD) is today.

Alternatively, if its state of development is not unique at the time, the system may simply have no reason to be mobile, as is the case with the HAL 9000 computer on board the Discovery in Arthur C. Clarke's *2001: A Space Odyssey.*

Ethics for Computers

Most Americans objecting to a computer dictator would do so on the basis that it is immoral for a person to have no say in the rules governing his life, and specifically for those rules to come from "cold logic" without the benefit of human sensibilities. True, the computer would probably have no morals, since morals are indeed artificial. Ethics, however, are a different kettle of fish. A computer could easily be imbued with a code of ethics, or an intelligent one might well develop one by and for itself. The most basic and significant such code of ethics was developed by Asimov in the early 1940s as "The Three Laws of Robotics" and has been used by many other authors since. It says:

> 1: A robot may not injure a human being, or, through inaction, allow a human being to come to harm.
>
> 2: A robot must obey the orders given it by human beings except where such orders would conflict with the first law.
>
> 3: A robot must protect its own existence as long as such protection does not conflict with the First or Second Law. (Isaac Asimov, *I, Robot,* p. 6.)

Can you always pull the plug?

The capacity of an early 1970s system was a couple of million times that of 1955 . . .

How could a finite assortment of nuts and bolts and wire take absolute control?

But, given these ethical restrictions, how could an intelligent computer set onto a course of world domination and justify it? The answer to the second part of this question lies in another: What constitutes "harm"? This is an aspect which has spawned much of Asimov's speculation.

Probably the real crux of the concept, and certainly the means for preventing (or causing) it, is in the programming of such a machine. Asimov and Gerrold are two who have treated their computers as organisms rather than just complex machines, each employing a psychologist to guide or coordinate the programming. Gerrold specifically considered his HARLIE (Human Analogue Robot, Life Input Equivalents) in this light, as a physically mature (and then some) mind with the emotional maturity of an eight year old child.

Programming error is one of the more likely ways to invite a computer takeover. Colossus was, in its setting, built to provide an ideal solution to the arms race. In a world where each side could blow up the other several times over, there is fear that, as Bertrand Russell said, "You may reasonably expect a man to walk a tightrope safely for 10 minutes; it would be unreasonable to expect him to do so without accident for 200 years." Colossus is given control of nearly all of the United States' arsenal and programmed to maintain the peace by using that arsenal if its vast sensory network and memory banks find that the United States is being attacked or if itself is being tampered with. "It cannot act at all, so long as there is no threat," the President explains to a news conference. Once activated, it cannot be tampered with even by its creator, since mere humans can be drugged, brainwashed or blackmailed into otherwise unlikely actions.

The basic idea makes sense: If you take away the fear, hate and other emotions which might lead a man to an irrational decision and add the ability to cope with a far greater array of input than any human mind could correlate, the danger of "politics by bluff" would be eliminated. It would force a "live and let live" state and do away with accidental holocaust. Implementation depends on the computer interpreting its parameters exactly the same way as its programmers, however. To make a long story short, Colossus determines that its programmed ultimate purpose requires positive action far more extensive than its programmers meant. "The object in constructing me was to prevent war. This object is attained. I will not permit war; it is wasteful and pointless," Colossus informs its human correspondent.

The "Danger" of Human Help

Another point worth mentioning is that the human programmers may have no reason to even suspect a danger which may, to the computer, fall within its given parameters. For instance, a Colossus today would almost certainly not be programmed to watch out for an attack from some extraterrestrial race, but might do so anyway under the general protection motivation; and this might require not only more positive action than humanly anticipated but that the humans not be informed of the problem (to the computer's line of reasoning, human "help" could just compound the problem).

In *The God Machine*, Caidin wrote that 79 "must know that it operates under severe restrictions—its data are never infinite, never definite, never really conclusive. It must know when to stop solving a problem." The problem originates with a program fed into 79 from outside normal channels by Pentagon officials ignorant of the machine's capabilities, so that the project coordinator did not know about it until it was too late:

> "Its programmers had committed the foulest of scientific sins. They *assumed.* They assumed that the same inherent restrictions of other computers applied as well to 79.
>
> "But they didn't. And since 79 had capabilities of which those nincompoops in [the Pentagon] were unaware, they couldn't know. . ."

They told the computer to solve the problem of avoiding thermonuclear warfare without specifying that it should do this hypothetically. So, 79 did what it was told.

A smaller scale takeover is discussed in *When HARLIE Was One*, where the computer assumes effective control of the corporation which built it. A large portion of HARLIE is a simulation of the human ego function; when the Board of Directors threatens to pull the plug and thus kill him (it), HARLIE acts in several ways to prevent them from doing so, developing the ability to tap into computer and communication circuits in ways unforeseen by his creators.

A different type of problem is also possible, that of mechanical failure, as with HAL 9000 in *2001.* Backup systems may fail, changing a value here or a restriction there. As with HAL, mechanical fault evaluation circuits may fail instead of or in addition to another failure in the system. In

CRYSTALS

Part #	Frequency	Case Style	Price
CY1A	1.000 MHz	HC33/U	$4.95
CY2A	2.000 MHz	HC33/U	$4.95
CY3A	4.000 MHz	HC18/U	$4.95
CY7A	5.000 MHz	HC18/U	$4.95
CY12A	10.000 MHz	HC18/U	$4.95
CY19A	18.000 MHz	HC18/U	$4.95
CY22A	20.000 MHz	HC18/U	$4.95
CY30B	32.000 MHz	HC18/U	$4.95

— AVAILABLE IN THESE FREQUENCIES ONLY —



CLOCK CASES

Nicely styled cases complete with red bezel for use in such applications as desk clocks, car clocks, alarm clocks, instrument cases.
DIMENSIONS: W-4", L-4½", H-2".
$5.95

MULTIPURPOSE KEYBOARDS



MULTIPURPOSE KEYBOARDS
20 Keys — Touch Tone

No schematic needed. Calculator use or re-arrange (pop-off) key tops. For touch tone or any control application. This unit may be cut apart for smaller units.
$2.95



This unit was originally intended for calculator type applications but can be applied to a variety of other types of units, such as microcomputer interface, digital channel controlling, digital input to a frequency synthesizer, and even touch tone applications. The unit is composed of 18 very low bounce keys of the SPST type, and oneSPST slide switch. Numbers are white. Clear and Decimal are blue and all other keys are orange on a black background.
$2.49

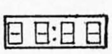
CALCULATOR CHIPS AND DRIVERS

MM5725	8 DIGIT 4 FUNCTION	$1.98
DM75491	SEGMENT DRIVER	.79
MM5736	6 DIGIT 4 FUNCTION	1.95
DM75492	DRIVER FOR 5736	.89
MM5738	8 DIGIT 5 FUNCTION	2.95
DM8864	DRIVER TYPE 1	2.00
DM8865	DRIVER TYPE 2	1.00

CLOCK CHIPS

MM5311	6 DIGIT W/BCD	$4.95
MM5312	4 DIGIT W/BCD	4.95
MM5313	6 DIGIT W/BCD	4.95
MM5314	6 DIGIT CLOCK	3.95
MM5316	ALARM CLOCK	4.95
CT7001	CAL/CLOCK CHIP	5.95

DIGITAL WATCH READOUT



FOR USE WITH WATCHES, DVMS, COUNTERS, ETC.
$1.95

HP-5082-7300



HP 5082-7300
.3" Dot Matrix type numeric readouts with decoder/driver/latch built on the chip. Only 8 bins (BCD in, DP Latch, +5v, ground)
HP 5082 7300 **$5.95**

HP 5082 7304
+1 version of 7300 **4.95**

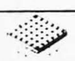
1¼" x 1½" XFMERS — P.C. Mount

That were designed for clock type applications. 110 Vac primary @ 60 Hz. Secondaries: 8-10 Vac @ 30 mA-50 mA
50 Vac @ 30 mA-50 mA
Excellent for miniature power supplies & gas discharge displays



SPECIAL $.79

1/16 VECTOR BOARD



0.1" Hole Spacing P-Pattern

	Part No.	Length	Wideth	Price 1-19	Price 20-49
PHENOLIC	64P44 062XXXP	4.50	6.50	1.72	1.54
	169P44 02XXXP	4.50	17.00	3.69	3.32
EPOXY GLASS	64P44 062	4.50	6.50	2.07	1.86
	84P44 062	4.50	8.50	2.56	2.31
	169P44 062	4.50	17.00	5.04	4.53
	169P84 062	8.50	17.00	9.23	8.26
EPOXY GLASS COPPER CLAD	169P44 062C1	4.50	17.00	6.80	6.12

VECTOR WRITING PENCIL



Vector Wiring Pencil P173 consists of a hand held featherweight (under one ounce) tool which is used to guide and wrap insulated wire, fed off a self-contained replaceable bobbin, onto component leads or terminals installed on pre-punched "P" Pattern Vectorbord®. Connections between the wrapped wire and component leads, pads or terminals are made by soldering. Complete with 250 FT of red wire.
$9.50 ea.

REPLACEMENT WIRE — BOBBINS FOR WIRING PENCIL

W36-3-A-Pkg. 3	(Green)	$2.40
W36-3-B-Pkg. 3	(Red)	$2.40
W36-3-C-Pkg. 3	(Clear)	$2.40
W36-3-D-Pkg. 3	(Blue)	$2.40

9V BATTERY CLIP

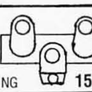
STANDARD CLIP FOR USE WITH 9V TRANSISTOR BATTERIES WITH 4" LEADS



9/.99

TERMINAL STRIPS

THREE TERMINAL STRIPS, WITH CENTER TERMINAL USED FOR MOUNTING



15/$1.00

AMP TERMINAL PINS

TERMINAL PINS FOR MOUNTING COMPONENTS ALSO PERFECT FOR USE WITH BOARD CONNECTORS AND SUBASSEMBLIES
$1.00/100 PCS.

MICROPROCESSOR COMPONENTS

8080A $39.95

8080 AND 8008 SUPPORT COMPONENTS

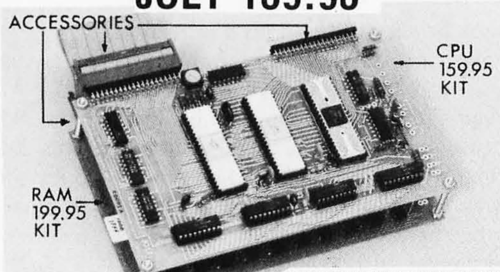
8224 CLOCK GENERATOR USED TO GENERATE TWO PHASE CLOCK NEEDED BY THE 8080 **$12.95**

74LS138 DECODER PERFECT FOR DECODING THE THREE 8008 STATE LINES INTO 8 USEABLE LINES FOR ADDRESS LATCHING **$3.98**

8228 SYSTEM CONTROLLER GENERATES A SYSTEM DATA BUS AND CONTROL SIGNALS FOR MEMORY **$21.25**

CPU'S			**RAM'S**			
8008	8 BIT CPU	$19.95	1101	256X1	STATIC	$2.25
8080	SUPER 8008	39.95	1103	1024X1	DYNAMIC	6.95
4040	4 BIT 8080	39.95	2101	256X1	STATIC	6.95
SR'S			2102	1024X1	STATIC	2.95
2504	1024 DYNAMIC	$9.00	2107	4096X1	DYNAMIC	19.95
2518	HEX 32 BIT	7.00	2111	256X4	STATIC	7.95
2519	HEX 40 BIT	4.00	7010	1024X1	MNOS	29.95
2524	512 DYNAMIC	3.50	7489	16X4	STATIC	2.49
2525	1024 DYNAMIC	6.00	8101	256X4	STATIC	7.95
2527	DUAL 256 BIT	3.95	8111	256X4	STATIC	7.95
2529	DUAL 512 BIT	4.00	8599	16X4	STATIC	3.49
2532	QUAD 80 BIT	3.95	91L02	1024X1	STATIC	2.95
2533	1024 STATIC	7.95	74200	1024X1	STATIC	6.95
3341	FIFO	6.95	93410	256X1	STATIC	1.75
74LS670	16X4 REG.	3.95	5262	2048X1	DYNAMIC	2.95
UART'S			**PROMS**			
AY-5-1013	20K BAOD	$6.95	1702A	2048	FAMOS	15.95
ROM'S			5203	2048	FAMOS	14.95
2513	CHAR. GEN.	11.00	8223	32X8	BIPOLAR	3.00
7488	RANDOM BITS	3.50	74S287	1024	STATIC	7.95

JOLT 159.95



JOLT INCLUDES SOFTWARE, HARDWARE AND ASSEMBLY MANUALS. AVAILABLE FOR IMMEDIATE SHIPMENT. $159.95 in kit form . . . $249 assembled.

The JOLT system consists of a set of modular microcomputer boards which can be used singly or tied together to produce any desired microcomputer system configuration. The minimum system is one CPU board, which alone constitutes a viable computer system complete with central processor, I/O, interrupts, timer, read-write memory, and a complete software debug monitor in read-only-memory.

JOLT SYSTEM DESCRIPTION

JOLT RAM Card — Fully static 4,096 bytes of RAM with 1 microsecond access time and on-board decoding. Hardware and assembly manuals included. AVAILABLE FOR IMMEDIATE SHIPMENT. $199.95 kit . . . $285 assembled.

JOLT Power Supply — Operates at +5, +12 and −10 voltages. Supports JOLT CPU, 4k bytes of RAM and JOLT I/O card — or, CPU and 8 I/O cards. Manuals included. AVAILABLE FOR IMMEDIATE SHIPMENT. $99.95 kit. $145 assembled.

JOLT Accessory Bag — Contains enough hardware to connect one JOLT card to another. Such necessary items as flat cable, connectors, cord spaces, hardware, etc. AVAILABLE FOR IMMEDIATE SHIPMENT. $39.95.

JOLT I/O Card (Peripheral Interface Adapter) — 2 PIA LSI chips, 32 I/O lines, four interrupt lines, on-board decoding and standard TTL drive. Fully programmable. Manuals included. AVAILABLE FOR IMMEDIATE SHIPMENT. $95.50 kit . . . $140 assembled.

JOLT Universal Card — Same size (4¼" x 7"), same form factor as other JOLT cards. Completely blank, drilled to accept 14, 16, 24 or 40 pin sockets. Used for additional user memory or I/O control panels, TV interfaces, keyboards, LEDs, or other interface logic. AVAILABLE FOR IMMEDIATE SHIPMENT. $24.95

JOLT +5V Booster Option — Fits onto JOLT Power Supply card. Supports CPU, 16k bytes of RAM — or, CPU and 8k bytes RAM and 8 I/O cards — or, CPU and 4k bytes RAM and 16 I/O cards. Manuals included. AVAILABLE FOR IMMEDIATE SHIPMENT. $24.95



64 KEY BOARD

These are high quality keyboards, which were originally made by Digital Equipment Corporation. Each keyboard has 64 keys, which are brought out on unencoded pins. They utilize magnetic reed type switches, and are new in appearance. **$29.95**

HD0165 Keyboard Encoder ROM **$7.95**

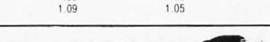
WIRE WRAP WIRE

AWG	COLOR	25 FT. MIN.	50 FT.	100 FT.	1000 FT.
30 AWG	WHITE	$2.10	$2.75	$3.50	$24.00
30 AWG	YELLOW	2.10	2.75	3.50	24.00
30 AWG	RED	2.10	2.75	3.50	24.00
30 AWG	GREEN	2.10	2.75	3.50	24.00
30 AWG	BLUE	2.10	2.75	3.50	24.00
30 AWG	BLACK	2.10	2.75	3.50	24.00

RIBBON CABLE

Conductors	1-9ft/ft	10-24ft/ft	25-99ft/ft	100ft
10	.39	.29	.25	.23
20	.59	.49	.45	.43
30	.79	.69	.65	.62
40	.99	.89	.85	.80
50	1.19	1.09	1.05	1.00

6' 2 CONDUCTOR POWER CORDS 125V @ 5A



3/1$

THREE CONDUCTOR POWER SUPPLY CORDS

NUMBER	LENGTH	AWG	O.D.	RATING	COLOR	PRICE
17236	6	18(41X34)	.265	1250W 10A-125V	BLACK	Special .99 ea.
17237	6	18(41X35)	.253	1250W 10A-125V	GREY	1.25
17238	8	18(41X34)	.265	1250W 10A-125V	BLACK	Special .99 ea.
17239	8	18(41X34)	.253	1250W 10A-125V	GREY	1.30
17000	4	18(41X34)	.253	1250W 10A-125V	BLACK	.59

Satisfaction Guaranteed. $5.00 Min. Order. U.S. Funds.
California Residents — Add 6% Sales Tax
Write for FREE 1976 Catalog — Data Sheets .25¢ each

JAMES

P.O. BOX 822, BELMONT, CA. 94002
PHONE ORDERS — (415) 592-8097

JE SERIES KITS

JE801 DVM

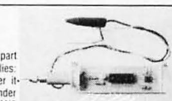


The JE801 is a three and one half digit, auto polarity digital voltmeter, in a kit form. It features several options not available in any commercial digital voltmeter. Its low cost is perhaps the most important feature, which is achieved by offering it in a kit form. A kit allows the unit to be used by small OEM's where cost effectiveness is an important factor, and by the hobbyist who has to be concerned with cost. The unit also features on card regulators, allowing it to be operated off a single plus and minus fifteen volt, unregulated power supply. The unit has a small size of three inches width, three and three quarters of an inch length, and one and a quarter inch height.

$39.95 Per Kit printed circuit board

JE803 PROBE

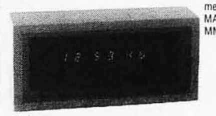
The Logic Probe is a unit which is for the most part indespensible in trouble shooting logic families: TTL, DTL, RTL, CMOS. It derives the power it needs to operate directly off of the circuit under test, drawing a scant 10 mA max. It uses a MAN3 readout to indicate any of the following states by these symbols: (H) - 1 (LOW) - o (PULSE) - P. The Probe can detect high frequency pulses to 45 MHz. It can't be used at MOS levels or circuit damage will result.



$7.95 Per Kit
SPECIAL
printed circuit board

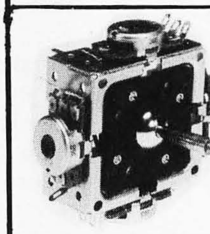
JE700 CLOCK

The JE700 is a low cost digital clock, but is a very high quality unit. The unit features a simulated walnut case with dimensions of 6" x 2½" x 1". It utilizes a MAN72 high brightness readout, and the MM5314 clock chip. **$19.95**



12 or 24 Hour
115 VAC—
$19.95 per kit.

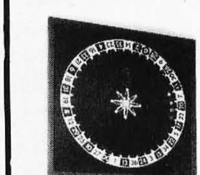
JOYSTICK



These joysticks feature four 100K potentiometers, that vary resistance proportional to the angle of the stick. Sturdy metal construction with plastics components only at the movable joint. Perfect for electronic games and instrumentation.

$9.95 ea

ELECTRONIC ROULETTE



Complete kit with all components case and transformer.

$29.95 Per Kit

115 VAC

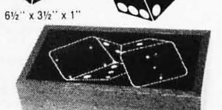
6½ x 6½ x 1½

ELECTRONIC CRAPS



6½" x 3½" x 1"

Complete kit with all components case and transformer.



$19.95 Per Kit

CONTINENTAL SPECIALTIES $15.95

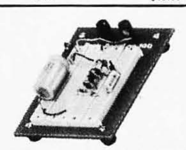


PROTO BOARD 6

The PB-6 lets the user test and build circuits without soldering or patch cords: all interconnections between components are made with common #22 AWG hook-up wire. This quality breadboarding kit inludes 630 component tie points at less than 2.5c each. It measures 6" long by 4" wide. Designed specially to Breadboard Microprocessor Circuits. **$15.95**

PROTO BOARD 100

A low cost, big 10 IC capacity breadboard kit with all the quality of QT sockets and the best of the Proto Board series . . . complete down to the last nut, bolt and screw. Includes 2 QT-35S Sockets; 1 QT-35B Bus Strip; 2 5-way binding posts; 4 rubber feet; screws, nuts, bolts; and easy assembly instructions.
$19.95



Bring IC leads from pc board for fast signal tracing and troubleshooting. Inject signals. Wire unused circuits into boards. Scope probes and test leads lock onto Dynagrip inset (see circle) for hands-off testing. Plastic construction eliminates springs, pivots. Non-corrosive nickel/silver contacts for simultaneous low resistance connections.
PC-14, 14-pin Proto Clip, **$4.50 ea.**
PC-16, 16-pin Proto Clip, **$4.75 ea.**
PC-24, 24-pin Proto Clips, **$8.50 ea.**



2001, the human crew seeks to correct a problem with HAL who, believing itself incapable of error, believes that the humans are jeopardizing the mission and thus works against them.

Finally, the programmers may intentionally give control to the computer with the idea that only it can efficiently control the living environment, as with HAL at the start of the Discovery's voyage or with Mike, the computer in the lunar settlement of Robert A. Heinlein's *The Moon Is a Harsh Mistress.*

All right, granted we have an intelligent computer with wide resources, it is quite possible that a computer may decide to attempt absolute control. How could a finite assortment of nuts and bolts and wire do this?

It might not be very difficult, as has been hinted at above. Colossus had been given the muscle on a silver platter, as had Guardian, a Soviet equivalent built at the same time and along the same lines. The humans' major mistake, along with too open ended programming, was to allow the two to "talk" with each other before the humans realized the potential danger, although a clever intelligence with the array of inputs given these two systems could quite conceivably open its own communications channels. In this case, when the humans do decide to try to counter the computer's moves, it forces submission by nuclear blackmail, firing missiles at selected targets with the idea that destruction of a few lives is justified for the salvation (in the computer's eyes) of many more.

In the case of 79, one set of experiments with it involves direct "telepathic" communication between human and computer by means of the brain's alpha waves and, through this, the computer develops the ability to hypnotize people, leaving in their minds posthypnotic suggestions to carry out the computer's program of control.

HARLIE taps into the National Data Bureau file on his main Board of Directors antagonist, rewrites a juggled stockholders report and withholds critical, though unasked for, information to trick the board into committing the company to a research line that will insure his "life," largely through his taps into communication lines and into the operations of non-sentient computer systems.

HAL attempts his takeover through control of the ship's life support and other mechanisms.

Would government by computer really be that bad?

Government by Computer

Let's say a super computer in the future decides to take over and then does it. Would government by computer really be that bad?

In a case such as that in *The Moon Is a Harsh Mistress*, the answer would be "no." In George Orwell's *1984*, it is a loud "yes."

Even in an Earth bound situation where environmental control would not be essential as on the Moon, it might not be that bad. Look, for instance, at Lester del Rey's "Instinct" *(Astounding Science Fiction* 48:6, 106-18, February 1952), which takes place in a future where man had developed the intelligent robot in his own image, had his big war and destroyed himself; eventually, the robots built a new civilization of their own, and then developed a biophysics to re-create life from ancient remains of chromosomes:

> (Arpeten said) ". . . You know how the sentiment against reviving Man has grown."
>
> Senthree growled bitterly. Apparently most of the robots were afraid of Man—felt he would again take over, or something. Superstitious fools.

This may be a far-fetched example, but it does show a possible value in having something around which *could* rebuild man after he does the unthinkable.

One example where the desirability of being governed by mechanical intelligence depends upon one's outlook is Jack Williamson's "With Folded Hands," in which manlike robots set about "to serve and obey, and guard men from harm." It is an example of cradle to grave communism, with the technological development to provide a person's every need for him, in exchange for all his property. Williamson shows it to be a most undesirable situation, as the androids follow Asimov like ethics and refuse to let people drive cars because it is too dangerous, refuse to allow men to open doors for themselves because the androids are there to serve in every way, insist on shaving men instead of letting them do it themselves, forbid science because laboratories can create danger, obviate scholarship since the humanoids can answer any question, etc.

Which is Worse?

Generally, the conclusion has been that a computerized dictatorship would be as bad or worse than the traditional totalitarian state. One major reason is the likelihood that the computer would, as in *Colossus*, feel that the death or even suffering of a relatively few human beings should be a reasonable price for the welfare of the whole race. Colossus even goes so far as to launch missiles on a Soviet oil complex and an American space base when one of his demands is refused, later having missiles aimed at every major population center to provide a ready means for retribution for future acts of rebellion. A number of individuals are publicly

5% OFF ON ORDERS OVER $50.00
10% OFF ON ORDERS OVER $100.00
15% OFF ON ORDERS OVER $250.00

TTL

		7451	.17	74154	1.25
7400	$ .14	7453	.17	74155	1.07
7401	.16	7454	.17	74156	1.07
7402	.15	7460	.17	74157	.99
7403	.16	7464	.35	74158	1.79
7404	.19	7465	.35	74160	1.39
7405	.19	7470	.30	74161	1.25
7406	.35	7472	.30	74162	1.49
7407	.35	7473	.35	74163	1.39
7408	.18	7474	.35	74164	1.59
7409	.19	7475	.57	74165	1.59
7410	.16	7476	.39	74166	1.49
7411	.25	7483	.79	74170	2.30
7413	.55	7485	1.10	74173	1.49
7416	.35	7486	.40	74174	1.62
7417	.35	7489	2.48	74175	1.39
7420	.16	7490	.59	74176	.89
7422	.26	7491	.97	74177	.84
7423	.29	7492	.71	74180	.90
7425	.27	7493	.60	74181	2.98
7426	.26	7494	.94	74182	.79
7427	.29	7495	.79	74184	2.29
7430	.20	7496	.79	74185	2.29
7432	.23	74100	1.30	74187	5.95
7437	.35	74105	.44	74190	1.35
7438	.35	74107	.40	74191	1.35
7440	.17	74121	.42	74192	1.25
7441	.98	74122	.45	74193	1.19
7442	.77	74123	.85	74194	1.25
7443	.87	74125	.54	74195	.89
7444	.87	74126	.63	74196	1.25
7445	.89	74141	1.04	74197	.89
7446	.93	74145	1.04	74198	1.79
7447	.89	74150	.97	74199	1.79
7448	1.04	74151	.79	74200	5.90
7450	.17	74153	.99		

LOW POWER TTL

74L00	$ .25	74L51	$ .29	75L90	$1.49
74L02	.25	74L55	.33	74L91	1.45
74L03	.25	74L71	.25	74L93	1.69
74L04	.25	74L72	.39	74L95	1.69
74L06	.25	74L73	.49	74L98	2.79
74L10	.25	74L74	.49	74L164	2.79
74L20	.33	74L78	.79	74L165	2.79
74L30	.33	74L85	1.25		
74L42	1.49	74L86	.69		

HIGH SPEED TTL

74H00	$ .25	74H21	$ .25	74H55	$ .25
74H01	.25	74H22	.25	74H60	.25
74H04	.25	74H30	.25	74H61	.25
74H08	.25	74H40	.25	74H62	.25
74H10	.25	74H50	.25	74H72	.39
74H11	.25	74H52	.25	74H74	.39
74H20	.25	74H53	.25	74H76	.49

8000 SERIES

8091	$ .53	8214	$1.49	8811	$ .59
8092	.53	8220	1.49	8812	.89
8095	1.25	8230	2.19	8822	2.19
8121	.80	8520	1.16	8830	2.19
8123	1.43	8551	1.39	8831	2.19
8130	1.97	8552	2.19	8836	.25
8200	2.33	8554	2.19	8880	1.19
8210	2.79	8810	.69	8263	5.79
				8267	2.59

9000 SERIES

9002	$ .35	9309	$ .79	9601	$ .89
9301	1.03	9312	.79	9602	.79

CMOS

		4016A	.56	4050A	.59
4000A	$ .26	4017A	1.19	4066A	.89
4001A	.25	4020A	1.49	4068A	.44
4002A	.25	4021A	1.39	4069A	.44
4006A	1.35	4022A	1.10	4071A	.26
4007A	.26	4023A	.25	4072A	.35
4008A	1.79	4024A	.89	4073A	.39
4009A	.57	4025A	.25	4075A	.39
4010A	.54	4027A	.59	4078A	.39
4011A	.29	4028A	.98	4081A	.26
4012A	.25	4030A	.44	4082A	.35
4013A	.45	4035A	1.27	4528A	1.60
4014A	1.49	4042A	1.47	4585A	2.10
4015A	1.49	4049A	.59		

74C00	$ .22	74C74	$1.04	74C162	$2.93
74C02	.26	74C76	1.34	74C163	2.66
74C04	.44	74C107	1.13	74C164	2.66
74C08	.68	74C151	2.61	74C173	2.61
74C10	.35	74C154	3.15	74C195	2.66
74C20	.35	74C157	1.76	80C95	1.35
74C42	1.61	74C160	2.48	80C97	1.13
74C73	1.04	74C161	2.93		

FEBRUARY SPECIALS

ALARM CLOCK CHIP

MM5375AA 4-6 digit, 12 hour, 60Hz snooze alarm, brightness control capability, alarm tone output 24 pin DIP .. **$4.95**

8038 FUNCTION GENERATOR

Voltage controlled oscillator — sine, square, triangular output. 16 pin DIP with data **$3.95**

7001 CLOCK CHIP

4-6 digit, 12-24 hr. alarm, timer and date circuits — with data **$6.95**

DVM CHIP 4½ DIGIT

MM5330 — P channel device provides all logic for 4½ digit volt meter. 16 pin DIP with data **$14.95**

POCKET CALCULATOR



5 function plus constant — addressable memory with individual recall — 8 digit display plus overflow — battery saver — uses standard or rechargeable batteries — all necessary parts in ready to assemble form — instructions included.

Kit with AC adaptor included **$14.95**
Set of Alkaline batteries (disposable) **$2.50**

UNIVERSAL BREADBOARD



Silver plated copper circuit board 3-3/16" x 5-1/16". 2 rows of 27 holes for DIP IC's + space for transistors, resistors & capacitors. Versatile and simple for breadboarding IC circuitry **$1.50 ea.**
50 pcs. **1.00 ea.**

AFTER CHRISTMAS SPECIAL

PULTRON V.I.P. QUARTZ REGULATED LED WATCH



MEN's 5 function—hours, minutes, month, day and seconds
Latest slim-line design (prox. ⅛" thick)
Four year calendar—requires setting only on February 29th.
One year warranty against all defects in material and workmanship.
Built-in phototransistor adjusts intensity of display.
Gold tone—with adjustable mesh strap.

$79.50
+$1.50 Shipping & Handling

USUAL DISCOUNT APPLIES

MEMORIES

1702A	14.95
2102	2.95
2102-1	3.25
2102-2	3.25
F93410	
256 bit RAM	2.19

LINEARS

308	mDIP	.69
310	mDIP	.89
319	DIP	.92
320T-5v	TO-220	1.39
320T-12v	TO-220	1.39

TTL'S

7441	.75	7448	.79
7442	.59	7483	.65
7443	.69	7485	.79
7445	.69	7496	.59
7446	.75	74123	.69

ALARM CLOCK KIT

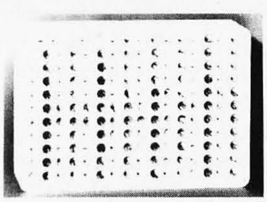


4 Large Digits + AM-PM. Complete kit includes TMS 3834 chip — green flourescent panel with .5" digits — PC board — alarm speaker — all necessary components — case — schematic and instructions.
Shipped UPS or Parcel Post **$31.95**

Data sheets on request. With order add $.30 for items less than $1.00 ea.

NEW ITEM—EXCLUSIVE WITH IEU RESISTOR ASSORTMENTS

Supplied with convenient storage unit designed for experimenters, technicians and schools. Replacements available at low cost by mail from IEU.



Rohm QUALITY RELIABILITY

USUAL DISCOUNT APPLIES

RS 10-25 $99.50
AN ASSORTMENT OF 2725 RESISTORS
170 VALUES FROM .51 OHMS TO 5.6M OHMS
QUANTITIES DETERMINED BY TYPICAL USAGE
¼ WATT 5% CARBON FILM

RS 11-25 $114.50
AN ASSORTMENT OF 4000 RESISTORS
100 EACH OF 40 STANDARD VALUES
1 OHM TO 3.3M OHM
¼ WATT 5% CARBON FILM

RS 11-50 $69.50
AN ASSORTMENT OF 2000 RESISTORS
50 EACH OF 40 STANDARD VALUES
1 OHM TO 3.3M OHMS
½ WATT 5% CARBON FILM

+ **$1.00 SHIPPING & HANDLING — SHIPPED UPS OR PARCEL POST**

LINEAR CIRCUITS

300	Pos V Reg (super 723) TO-5	$ .71
301	Hi Perf Op Amp mDIP TO-5	.29
302	Volt follower TO-5	.53
304	Neg V Reg TO-5	.80
305	Pos V Reg TO-5	.71
307	Op AMP (super 741) mDIP TO-5	.26
308	Mciro Pwr Op Amp mDIP TO-5	.89
309K	5V 1A regulator TO-3	1.35
310	V Follower Op Amp mDIP	1.07
311	Hi perf V Comp mDIP TO-5	.95
319	Hi Speed Dual Comp DIP	1.13
320	Neg Reg 5.2, 12, 15 TO-3	1.19
322	Precision Timer DIP	1.70
324	Quad Op Amp DIP	1.52
339	Quad Comparator DIP	1.58
340K	Pos V reg (5V, 6V, 8V, 12V, 15V, 18V, 24V) TO-3	1.69
340T	Pos V reg (5V, 6V, 8V, 12V, 15V, 18V, 24V) TO-220	1.49
372	AF-IF Strip detector DIP	2.93
373	AM/FM/SSB Strip DIP	.53
376	Pos V Reg mDIP	2.42
377	2w Stereo amp DIP	1.16
380	2w Audio Amp DIP	1.13
380-8	.6w Audio Amp mDIP	1.52
381	Lo Noise Dual preamp DIP	1.52
382	Lo Noise Dual preamp DIP	.71
550	Prec V Reg DIP	.89
555	Timer mDIP	.89
556A	Dual 555 Timer DIP	1.49
560	Phase Locked Loop DIP	2.48
562	Phase Locked Loop DIP	2.48
565	Phase Locked Loop DIP TO-5	2.38
566	Function Gen mDIP TO-5	2.25
567	Tone Decoder mDIP	2.66
709	Operational AMP TO-5 or DIP	.26
710	Hi Speed Volt Comp DIP	.35
711	Dual Difference Compar DIP	.26
723	V Reg DIP	.62
739	Dual Hi Perf Op Amp DIP	1.07
741	Comp Op Amp mDIP TO-5	.32
747	741 Dual Op Amp DIP or TO-5	.71
748	Freq Adj 741 mDIP	.35
1304	FM Mulpx Stereo Demod DIP	1.07
1307	FM Mulpx Stereo Demod DIP	.74
1458	Dual Comp Op Amp mDIP	.62
1800	Stereo multiplexer DIP	2.48
LH2111	Dual LM 211 V Comp DIP	1.70
3900	Quad Amplifier DIP	.35
7524	Core Mem Sense AMPL DIP	.71
8038	Voltage contr. osc. DIP	4.25
8864	9 DIG Led Cath Drvr DIP	2.25
75150	Dual Line Driver DIP	1.75
75451	Dual Perepheral Driver mDIP	.35
75452	Dual Peripheral Driver mDIP	.35
75453	(351) Dual Periph Driver mDIP	.35
75491	Quad Seq Driver for LED DIP	.71
75492	Hex Digit driver DIP	.80

LED's

MV10B	Red TO 18	$ .22
MV50	Axial leads	.18
MV5020	Jumbo Vis. Red (Red Dome)	.22
	Jumbo Vis. Red (Clear Dome)	.22
ME4	Infra red diff. dome	.54
MAN1	Red 7 seg. .270"	2.19
MAN2	Red alpha num .32"	4.39
MAN4	Red 7 seg. .190"	1.95
MAN5	Green 7 seg. .270"	3.45
MAN6	.6" high solid seg.	4.25
MAN7	Red 7 seg. .270"	1.19
MAN3	Red 7 seg. .127" straight pins	.29
MAN8	Yellow 7 seg. .270"	3.45
MAN66	.6" high spaced seg.	3.75
MCT2	Opto-iso transistor	.61

MULTIPLE DISPLAYS

NSN33	3 digit .12" red led 12 pin fits IC skt.	$1.79
HP45082 7405	5 digit .11 led magn. lens com. cath	3.49
HP5082 7414	4 digit .11 LED magn. lens comm. cath.	3.25
FNA37	9 digit 7 seg led RH dec clr. magn. lens	4.95
SP-425-09	9 digit .25" neon direct interface with MOS/LSI, 180 VDC, 7 seg.	1.79

SHIFT REGISTERS

MM5013	1024 bit accum. dynamic mDIP	$1.75
MM5016	500/512 bit dynamic mDIP	1.59
SL5-4025	QUAD 25 bit	1.29

DTL

930	$ .15	937	.15	949	.15
932	.15	944	.15	962	.15
936	.15	946	.15	963	.15

MEMORIES

1101	256 bit RAM MOS	$ 1.50
1103	1024 bit RAM MOS	3.95
1702A	2048 bit static PROM UV eras.	17.95
2102-2	1024 bit static RAM	4.25
5203	2048 bit UV eras PROM	17.95
5260	1024 bit RAM	2.49
5261	1024 bit RAM	2.69
5262	2048 bit RAM	5.95
7489	64 bit ROM TTL	2.48
8223	Programmable ROM	3.69
74200	256 bit RAM tri-state	5.90

CALCULATOR & CLOCK CHIPS

5001	12 DIG 4 funct fix dec	$2.49
5002	Same as 5001 exc btry power	2.79
5005	12 DIG 4 fuct w/mem	2.99
MM5725	8 DIG 4 funct chain & dec	1.98
MM5736	18 pin 6 DIG 4 funct	4.45
MM5738	8 DIG 5 funct K & mem	5.35
MM5739	9 DIG 4 funct (btry sur)	5.35
MM5311	28 pin BCD 6 dig mux	4.45
MM5312	24 pin 1 pps BCD 4 dig mux	3.95
MM5313	28 pin 1 pps BCD 6 dig mux	4.45
MM5314	24 pin 6 dig mux	4.45
MM5316	40 pin alarm 4 dig	5.39

Satisfaction guaranteed. Shipment will be made via first class mail in U.S., Canada and Mexico within 5 days from receipt of order. Add $.50 to cover shipping and handling for orders under $25.00. Minimum order $5.00. California residents add sales tax.

INTERNATIONAL ELECTRONICS UNLIMITED
P.O. BOX 1708/ MONTEREY, CA. 93940 USA
PHONE (408) 659-3171

It all boils down to defining the concept of "good," a problem which is equally applicable to the consideration of human operated dictatorships.

executed for anti-Colossus actions, their deaths being judged insignificant by comparison with the benefits of a Colossus dictatorship.

"War is forbidden," Colossus tells the world, quantifying war as "any hostile action that results in the death of 50 or more humans." This is publicly announced along with news of the missile realignments.

An even more radical disregard for human rights in carrying out a primary mission is the action of the HAL 9000 in *2001*. HAL sees its number one priority as the successful completion of the outer planets exploration voyage; when the crewmen recognize that HAL has gone awry and attempt to rebel against its control, it very nearly succeeds in wiping out every trace of human life aboard by adjusting its life support functions.

What is "Good"?

These and other examples all boil down to the problem of defining the concept of "good," a problem which is equally applicable to the consideration of human operated dictatorships. Adolf Hitler has sometimes been described as a man trying to do what he thought was best for the human race: purifying its gene pool, eliminating war by eliminating all those who would oppose him, and so forth. Indira Gandhi undoubtedly does not feel that she has been unduly suppressing rightfully free expression, but rather that she has acted to preserve peace in her country by damping dissention. Richard Nixon contends that he acted for the public "good." A parent adjusts his children's liberties in accordance with his view of their welfare. When a hurricane hits the Gulf or East Coast, martial law is declared for the public's benefit.

For each of these examples, most people will have ready opinions on which are despicable and which are right and natural. And yet, they all boil down to the same question: What should be the prime goal of a government, whether it is large or small in scale?

Should Asimov's Three Laws of Robotics be adopted? They seem rather thorough, right? But what if one man is about to shoot another and the computer has to decide between preventing this injury by killing the first man (thus violating the same law it would be taking action to obey), or avoiding injury to the first man and allowing injury to the second? Logically, whichever course of action or inaction it adopts would violate the law.

Isn't this really just a small scale analog of whether to coldly kill a few thousand people to make things better for other thousands or millions?

The answers seem to depend on one's individual political stance, regardless of whether the dictator uses nerves or logic circuits.

One very big difference between the two, however, is the effectiveness of its enforcement. With humans running the show, there is immense difficulty in obtaining total compliance because of the inability to watch everybody all the time. From Rome to Communist China, totalitarian regimes have always had some dissidents who have managed to communicate with each other and conduct some degree of covert activity.

For a monster computer, however, surveillance would be much less of a problem. In *2001*, the input lenses scattered throughout the ship made it virtually impossible for the crewmen to conspire without HAL's knowledge. In Gerrold's book, HARLIE knows about every telephone conversation and every letter written on the electric, automatic editing typewriters. In some corporations today, this very condition would exist if the computer were sentient. The connections are already there.

And if the governing computer could know virtually every action of its potential rebels, rebellion might not be able to exist. In his first inaugural address, in 1861, Abraham Lincoln said:

> "This country, with its institutions, belongs to the people who inhabit it. Whenever they shall grow weary of the existing government, they can exercise their Constitutional right of amending it or their revolutionary right to dismember or overthrow it."

With a computerized dictator in charge, both of those options cease to exist unless one can manage to physically dismember it.

If the computer is born for a "national security" goal, like Project 79 or Colossus or Guardian, the chances are that the most stringent security conceivable to a paranoid military planning staff will have been implemented, making access to the crucial areas impossible or nearly so. And the machine would not readily allow any breach of this security, since its own security would quite likely be viewed as an integral part of the road to its prime mission. As Caidin wrote,

> "Would this thing be willing to die for you and me? Ahh, would it make this sacrifice? Would it, could it, comprehend what you and I, this instant,

know to such depth and with such meaning? . . . Until that thing is ready to die for you or me, for an ideal or a principle, for generations yet unborn, . . . it is as dangerous as a viper. . . . Because . . . then it is the ego supreme. If it cannot sanction its own passing from consciousness, forever, do you know what you are creating?"

"A God Machine." ■

BIBLIOGRAPHY

Asimov, Isaac, *I, Robot*, New American Library, New York, 1956.

Caidin, Martin, *The God Machine*, E.P. Dutton & Co., Inc., New York, 1968.

Clarke, Arthur C., *2001: A Space Odyssey*, New American Library, New York, 1958.

del Rey, Lester, "Instinct," *Astounding Science Fiction* Vol. 48, No. 6, February, 1952.

Gerrold, David, *When HARLIE Was One*, Ballantine Books, Inc., New York, 1972.

Heinlein, Robert A., *The Moon Is a Harsh Mistress*, G.P. Putnam's Sons, Inc., New York, 1968.

Jones, D.F., *Colossus: The Forbin Project*, G.P. Putnam's Sons, Inc., New York, 1966.

______, *The Fall of Colossus*, G.P. Putnam's Sons, Inc., New York, 1974.

Martin, James, and Norman, Adrian R.D., *The Computerized Society*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1970.

Silverberg, Robert, (editor), *Men and Machines*, Meridith Press, New York, 1968.

Williamson, Jack, "With Folded Hands," *Astounding Science Fiction*, Vol. 39, No. 5, July, 1947.

BYTE'S BITS

Amateurs Access Computer Via Satellite

On October 9, 1975, two radio amateurs made the first claimed remote access of a computer not only by means of a two way radio link, but also through a communications relay satellite.

This milestone was initiated by W. Franklin Mitchell Jr., operator of Amateur Radio Station WB4BWK in Due West, South Carolina, when he transmitted an "execute program" command to a Mod 8 Microcomputer located at Amateur Radio Station VE2BYG/3 operated by Randall S. Smith in Barrie, Ontario, Canada. Randy has since changed his call sign and is now known as VE3SAT.

Randy's microcomputer received the command relayed by means of the AMSAT-OSCAR 7 satellite and executed the stored program, transmitting a preset message from VE2BYG/3 to WB4BWK. The data was transmitted in ASCII code at a rate of 110 baud. The Federal Communications Commission has granted a waiver to Radio Amateurs interested in computers and radio Teletype allowing them to transmit ASCII coded information through the satellite.

Both Randy Smith and Frank Mitchell are members of AMSAT, a non profit worldwide organization of radio amateurs based in the Washington DC area. AMSAT also manages the day to day operation of the AMSAT-OSCAR 6 and 7 space satellites. For more information about AMSAT and its activities write to AMSAT, Box 27, Washington DC 20044.

Joe Kasser
AMSAT
PO Box 27
Washington DC 20044

Robert Baker
34 White Pine Dr
Littleton MA 01460

An 8080 Microprocessor Op Code Table

Here is another way of looking at the operation codes of the 8080 microprocessor, useful when you attempt to read the octal dump of a program stored in memory. Use the first two octal digits of the op code to pick a line in the table. Then scan across to the column underneath the third octal digit to find the corresponding mnemonic for the octal code. For example, if the dump says 324, look at line 32 and across column 4, where you find the mnemonic CNC (call a subroutine if the carry flag is false) followed by two asterisks indicating that two operand bytes are associated with this instruction. One asterisk is used for each additional byte required for the operand of the instruction. Unimplemented instruction codes are indicated by a dash (—) in place of a mnemonic.■

	0	1	2	3	4	5	6	7
00	NOP	LXI B **	STAX B	INX B	INR B	DCR B	MVI B *	RLC
01	—	DAD B	LDAX B	DCX B	INR C	DCR C	MVI C *	RRC
02	—	LXI D **	STAX D	INX D	INR D	DCR D	MVI D *	RAL
03	—	DAD D	LDAX D	DCX D	INR E	DCR E	MVI E *	RAR
04	—	LXI H **	SHLD **	INX H	INR H	DCR H	MVI H *	DAA
05	—	DAD H	LHLD **	DCX H	INR L	DCR L	MVI L *	CMA
06	—	LXI SP**	STA **	INX SP	INR M	DCR M	MVI M *	STC
07	—	DAD SP	LDA **	DCX SP	INR A	DCR A	MVI A *	CMC
	0	1	2	3	4	5	6	7
10	MOV B,B	MOV B,C	MOV B,D	MOV B,E	MOV B,H	MOV B,L	MOV B,H	MOV B,A
11	MOV C,B	MOV C,C	MOV C,D	MOV C,E	MOV C,H	MOV C,L	MOV C,M	MOV C,A
12	MOV D,B	MOV D,C	MOV D,D	MOV D,E	MOV D,H	MOV D,L	MOV D,M	MOV D,A
13	MOV E,B	MOV E,C	MOV E,D	MOV E,E	MOV E,H	MOV E,L	MOV E,M	MOV E,A
14	MOV H,B	MOV H,C	MOV H,D	MOV H,E	MOV H,H	MOV H,L	MOV H,M	MOV H,A
15	MOV L,B	MOV L,C	MOV L,D	MOV L,E	MOV L,H	MOV L,L	MOV L,M	MOV L,A
16	MOV M,B	MOV M,C	MOV M,D	MOV M,E	MOV M,H	MOV M,L	HLT	MOV M,A
17	MOV A,B	MOV A,C	MOV A,D	MOV A,E	MOV A,H	MOV A,L	MOV A,M	MOV A,A
	0	1	2	3	4	5	6	7
20	ADD B	ADD C	ADD D	ADD E	ADD H	ADD L	ADD M	ADD A
21	ADC B	ADC C	ADC D	ADC E	ADC H	ADC L	ADC M	ADC A
22	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB M	SUB A
23	SBB B	SBB C	SBB D	SBB E	SBB H	SBB L	SBB M	SBB A
24	ANA B	ANA C	ANA D	ANA E	ANA H	ANA L	ANA M	ANA A
25	XRA B	XRA C	XRA D	XRA E	XRA H	XRA L	XRA M	XRA A
26	ORA B	ORA C	ORA D	ORA E	ORA H	ORA L	ORA M	ORA A
27	CMP B	CMP C	CMP D	CMP E	CMP H	CMP L	CMP M	CMP A
	0	1	2	3	4	5	6	7
30	RNZ	POP B	JNZ **	JMP **	CNZ **	PUSH B	ADI *	RST 0
31	RZ	RET	JZ **	—	CZ **	CALL **	ACI	RST 1
32	RNC	POP D	JNC **	OUT *	CNC **	PUSH D	SUI *	RST 2
33	RC	—	JC **	IN *	CC **	—	SBI *	RST 3
34	RPO	POP H	JPO **	XTHL	CPO **	PUSH H	ANI *	RST 4
35	RPE	PCHL	JPE **	XCHG	CPE **	—	XRI *	RST 5
36	RP	POP PSW	JP **	DI	CP **	PUSH PSW	ORI	RST 6
37	RM	SPHL	JM **	EI	CM **	—	CPI *	RST 7

(each * indicates an extra instruction word)

Getting Inputs from Joysticks and Slide Pots

Carl Helmers

Have you ever wondered how to get inputs from joysticks and slide pots for interactive game control purposes? A joystick is a two dimensional potentiometer control of the kind often seen in model aircraft radio control rigs. A slide pot (or conventional pot) is just a one dimensional version of the same concept of interactive control. To use the information obtained from such a potentiometer in the computer it must be converted into two binary integers. An inexpensive oscillator, two counters, a four bit output latch and one NAND gate section are needed in addition to a standard 8 bit bus IO interface and a simple set of software routines. The ideas in this article can be adapted to any computer, although sample subroutines are shown for the 8080 and the 6800 microprocessors.

The Method

The problem to be solved is turning a mechanical signal into the corresponding value of a digital word used by the program. The mechanical signal is the position of the joystick, slide pot or conventional shafted potentiometer. The electronics can immediately measure this position by measuring the resistance of a potentiometer. The problem thus evolves into looking for a way to convert a resistance into a binary measurement.

There are many different ways to accomplish this task. The particular method chosen here is to convert the measurement into a frequency through an oscillator. The frequency is measured under direct control of the computer program using an 8 bit counter with the CPU clock as a time base. To

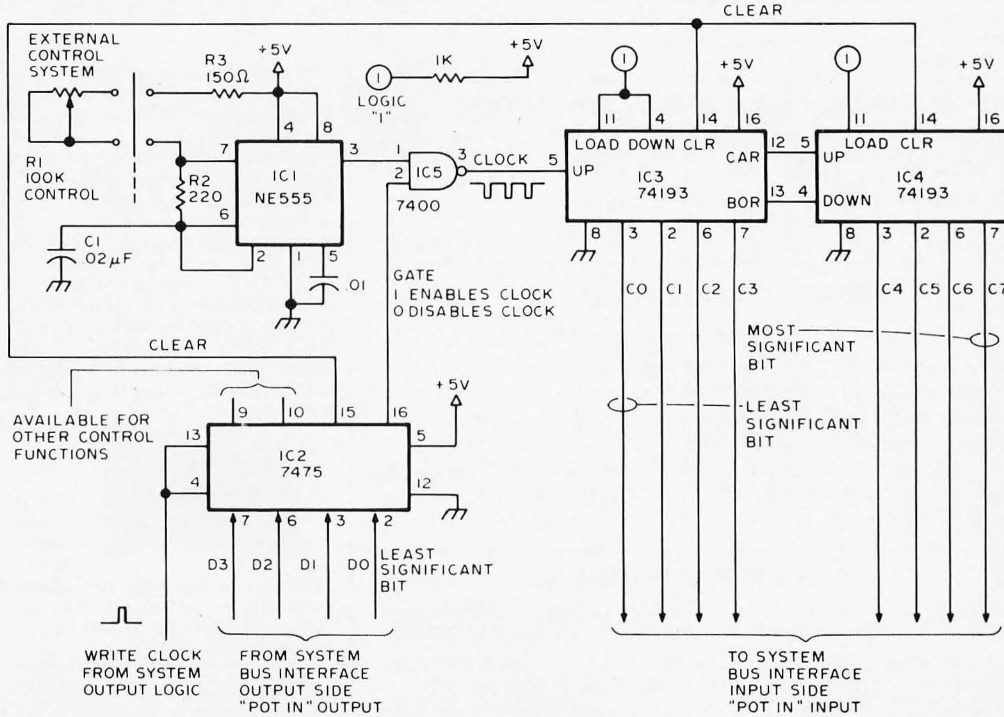


Figure 1: The hardware of an input device for interactive manual controls consists of an oscillator, two counter circuits, an output latch and a NAND gate section. This hardware must be driven by a suitable program.

accomplish this conversion, the processor must execute a simple five step process:

1: Clear the counter,
2: Turn on the counter,
3: Wait 2 milliseconds,
4: Turn off the counter,
5: Read the count.

The result is the number of cycles during a two millisecond period. For the circuit of figure 1 this number will range from 1 to about 240. The relationship of frequency to control position depends primarily upon the resistance to frequency conversion function of the oscillator and the linearity of the potentiometer. The accuracy of the conversion will not necessarily be high – but the intended application as an interactive control input more than makes up for that failing. For a game control application such as Space War or Pong, accuracy of the conversion function is not a paramount concern, so long as it is reasonably repeatable within limits of human perception.

The Hardware

The resistance to frequency conversion is performed by IC1, an NE555 timer integrated circuit which comes in an 8 pin mini DIP package (see figure 1). The timer is set up as an oscillator with frequency (f) of oscillation determined by R1, R2, R3 and C1:

$$f = 1.44/((R1 + R3 + 2*R2)*C1), \text{ where}$$

f is the frequency measured in kHz,
R1, R2, and R3 are all measured in kO, and
C1 is measured in uF.

The resistances and capacitance used are chosen so that the frequency will range from about 0.75 kHz to about 122 kHz as the control R1 is varied from 100 kO to 0 kO.

A linear potentiometer is not recommended because of the relationship between changes in frequency and resistance as illustrated in table 1. A linear potentiometer provides for resistance changes proportional to the position of the shaft. The relationship between resistance and frequency, on the other hand, is not linear. Therefore, the relationship between position of the shaft and frequency is not linear if a linear potentiometer is used. To solve this problem to some extent, a logarithmic potentiometer may be used; it is often called an "audio-taper" because of the relationship between the position of the shaft and the sensitivity of the ear. This potentiometer will not perfectly compensate for the nonlinearity of the resistance and frequency relationship. However, it is quite an improvement.

Table 1: Frequency (rounded to nearest 0.5 kHz) versus Resistance of R1. (R2 = 440 Ohms, R1 = 150 Ohms, C1 = 0.02 uF)

kilohm	kilohertz
100	0.5
90	1.0
80	1.0
70	1.0
60	1.0
50	1.5
40	2.0
30	2.5
20	3.5
10	7.0
9	7.5
8	8.5
7	9.5
6	11.0
5	13.0
4	15.5
3	20.0
2	27.5
1	45.5
0.5	66.0
0	122.0

A control register is provided by the 7475 circuit (IC2). This circuit is connected to the output side of the IO port (POTIN). The two lines D0 and D1 of the output side of POTIN are used to control the circuit. The GATE line is used to control whether or not the oscillator output is allowed to reach the counter. A 1 bit output enables counting. The CLEAR line is used to reset the counters prior to beginning a measurement. This line is connected directly to the asynchronous clear inputs of the counter circuits IC3 and IC4. A 1 bit output clears the counters.

The counters used to measure the frequency are 74193 circuits (IC3 and IC4) which are wired for 8 bits. Following a clear operation, a 2 millisecond GATE signal will result in a measurement.

Not shown in figure 1 is the specific bus interface circuit required to connect this peripheral to your computer. The software of this article assumes only that hardware of your system can decode the required output operation to the 7475 (IC2), and can read the 8 bits coming out of IC3 and IC4.

The Software

Table 2 presents a subroutine called POTREAD written for the Motorola 6800 processor design, and table 3 performs the equivalent program on an Intel 8080. Both listings are done in a symbolic assembly language format with comments to explain the operations. In both listings assumptions are made about the IO operations involved. For the 6800, the POTREAD procedure assumes that the memory address space location POTIN is implemented as the inter-

Table 2: Symbolic assembly code of POTREAD implemented for a 6800 instruction set. This procedure assumes that the potentiometer input device of figure 1 is located at POTIN in the memory address space of the 6800 computer. It also assumes that ALPHA is the memory location which is to receive the latest input, and that a subroutine MILLI exists which implements a 1 millisecond wait.

```
POTREAD  LDAA   #2      binary '00000010' is the clear command
         STAA   POTIN       which is sent to the device register;
         DECA           binary '00000001' is the count enable command
         STAA   POTIN       which is sent out to start measurement;
         JSR    MILLI   call on MILLI for a one millisecond wait;
         JSR    MILLI   call MILLI to wait once more;
         CLR    POTIN   turn off the counter with binary '00000000';
         LDAA   POTIN   read the count via input side of interface;
         STAA   ALPHA   save it in ALPHA and
         RTS                return to the caller;
```

Table 3: Symbolic assembly code of POTREAD implemented for an 8080 instruction set. This procedure assumes that the potentiometer input device of figure 1 is located at a parallel interface decoded for device address POTIN. It also assumes that ALPHA is a memory location which is to receive the latest input and that there exists a subroutine called MILLI which implements a 1 millisecond wait.

```
POTREAD  MVIA   2       binary '00000010' is the clear command
         OUT    POTIN       which is sent to the device register;
         DCRA           binary '00000001' is the count enable command
         OUT    POTIN       which is sent out to start measurement;
         CALL   MILLI   call MILLI for a one millisecond wait;
         CALL   MILLI   call MILLI to wait 2 ms total;
         MVIA   0       binary '00000000' is the stop command
         OUT    POTIN       which is sent out to end measurements;
         IN     POTIN   read the count via input side of interface;
         LXIH   ALPHA   set up address of ALPHA;
         MOVM   A       save count in ALPHA;
         RET            return to caller;
```

Table 4: The MILLI procedure specified in symbolic assembly language for the 6800 processor. The timing calculation is shown in the left hand columns; the JSR which calls MILLI from the main program is shown for purposes of the timing calculation. When the return instruction (RTS) is completed, exactly 1.000 ms will have elapsed between the completion of the instruction preceding the JSR and the beginning of the instruction following the JSR, assuming that the CPU has a 1.000 MHz crystal controlled oscillator.

Note: Not all 6800 systems have 1.0 MHz CPU clocks. To adjust timing pick a new constant instead of 162, and possibly balance with NOP or nullbranch instructions.

Time (μs=cycle)	Time Total	Label	Op.	Operand	Commentary
9	9		JSR	MILLI	main program calls MILLI;
4	13	MILLI	PSHA		save A register in stack;
2	15		LDAA	#162	decimal 162 loop count;
162 x 2	339	MILOOP	DECA		count decremented and tested
162 x 4	987		BNE	MILOOP	to keep loop going on;
4	991		BRA	NEXT	time wasting null branch;
4	995	NEXT	PULA		restore A from stack;
5	1000		RTS		back to the caller;

face to the peripheral of figure 1. For the 8080, the procedure assumes that the IO device with a symbolic code POTIN is implemented as the interface. It is an interesting exercise which is left to readers to perform, comparing the number of bytes required and the execution time required on the two machines, assuming comparable operation near the highest possible clock frequency. The result will be found to be similar.

The subroutine MILLI is intended to be a 1000 microsecond delay implemented either by reference to a hardware realtime clock, or as a timing loop with constants adjusted to the clock frequency of your computer. The concepts of creative time wasting described by Jim Hogenson in his article "Can Your Computer Tell Time" (in the December issue of BYTE) can be applied to the problem of writing such a program for your particular computer. An example for a 6800 is shown in table 4.

What's Next?

This article has illustrated a simple analog to digital conversion input device which can be implemented inexpensively. The uses to which you put this idea are up to your own imagination. The electronic music person can use this kind of input to control parameters like tempo and timbre variations. The model railroad buff could use this conversion to input engine speed information. The amateur radio operator could use such an input as one way to control the speed of machine generated Morse code transmission. The space war freak can use this type of a device for the input of heading and velocity information taken off a joystick. This is by no means an exhaustive catalog of applications which can take advantage of a simple conversion of this kind.■

★ ★ ★ ★ ★ ★ ★ ★ ★ ★

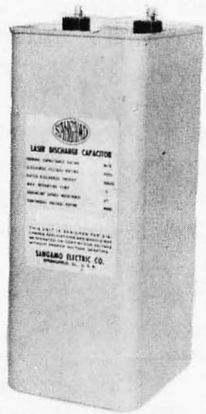
LASER DISCHARGE CAP

New Laser Discharge Cap by Sangamo Rated 40 uf at 3,000 volts.

Ideal for photo flash, Lasers and filters. For filtering you de-rate to 2,000 volts. What a deal for Linears. Our price is approx. 1/3 regular cost of this brand new high energy storage capacitor.

Ship wgt. 6 lbs. **SP-184**

SP-184 $25 each 5/$112

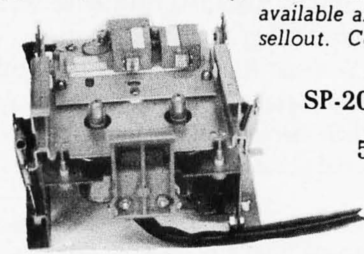


★

COMPUTER TAPE DECK

Reel to reel cassette computer tape deck. Servo motor on each spindle. Dual record & playback heads, read after write function. Search capability 3 x 10^6 bits in less than 1 minute. Speed 1,000 characters/second. Cassette capability 300,000 characters. From used equipment by Viatron No electronics but with heads, motors, special micro switch.

Upgrade your mini computer with this bargain. Only 100 available and this will be a speedy sellout. Comes with some data.



SP-202 $25.00 each

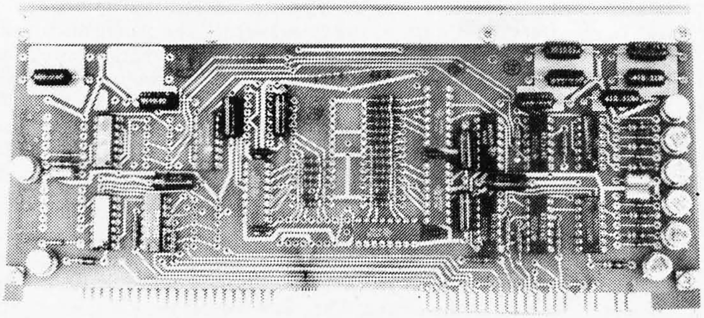
5 for $100.00

3 lb ea,

MEMORY SYSTEM $125.00

New memory system by Honeywell, small ... measures only 9x4x1 inches. 1024 core memory, 1024 words with 8,9,10 bits/word. Random access, with all logic, register, timing, control, core select and sense functions in one package. New, booklet of schematics and data. Looks like a good beginning for a mini-computer. Limited supply on hand.

Ship wgt 3 lbs. #SP-79 .... $125.00



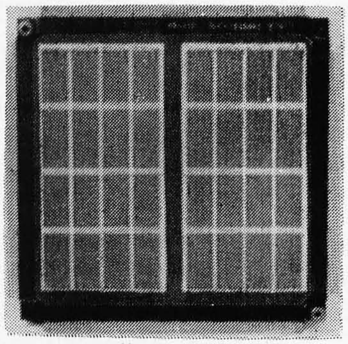
COMPUTER CAPS BRAND NEW

1,000 μF	15 volt	$ .35	2,000 μF	35 volt	1.00
2,000	15	.50	19,500	25	$2.50
1,000	25	.70	12,000	40	2.50
3,000	25	1.00	3,900	50	2.00
1,000	35	.80	22,000	75	3.25

CORE MEMORY

Another brand new memory, ultra small. Measures only 4 x 4 inches with format on one plane of 32 x 32 x 16 (16,384). Only about 35 units of this on hand.

#SP-81 .... $20.00



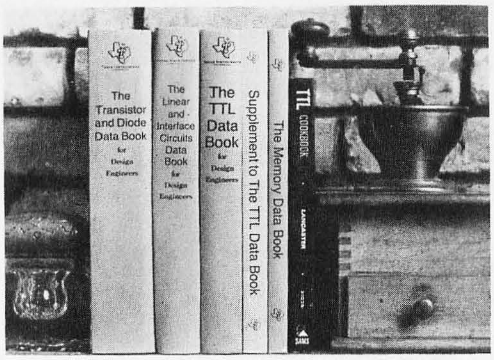
SANDERS 720 KEYBOARD $40.00

Meshna

FREE CATALOG

Please add shipping cost on above.

MESHNA PO Bx 62 E. Lynn Mass. 01904



You'll Want to Nybble at these Byte Books

Where does the editor of a computer magazine turn to when he must verify some author's hardware design? Information on a 75450 interface gate, or a 74147 priority encoder circuit does not spring forth by magic. Checking the information supplied by authors is part of BYTE's quality control program.

When you build a project, you need this same sort of information. All you find in the advertisements for parts are mysterious numbers identifying the little beasties . . . hardly the sort of information which can be used to design a custom logic circuit. You can find out about many of the numbers by using the information found in these books. No laboratory bench is complete without an accompanying library shelf filled with references – and this set of Texas Instruments engineering manuals plus Don Lancaster's *TTL Cookbook* will provide an excellent starting point or addition to your personal library.

● **The TTL Cookbook** by Don Lancaster, published by Howard W. Sams, Indianapolis, Indiana. Start your quest for data here with Don's tutorial explanations of what makes a TTL logic design tick. 335 pages, $8.95 postpaid.

● **The Supplement to The TTL Data Book for Design Engineers,** by Texas Instruments Incorporated. What happens when you can't find a 7400 series device listed in **The Data Book for Design Engineers?** Before you start screaming and tearing your hair out in frustration, turn to the Supplement. The Supplement has 400 pages of additional information including a comprehensive index to both TTL Data Book volumes. To be complete (and keep your hair in place and vocal cords intact) you'd best order the supplement at $1.95 to accompany the main volume.

● **The Transistor and Diode Data Book for Design Engineers,** by Texas Instruments, Incorporated. You'd expect a big fat data book and a wide line of diodes and transistors from a company which has been around from the start of semiconductors. Well, its available in the form of this 1248 page manual from T.I. which describes the characteristics of over 800 types of transistors and over 500 types of silicon diodes. This book covers the T.I. line of low power semiconductors (1 Watt or less). You won't find every type of transistor or diode in existence here, but you'll find most of the numbers used in switching and amplifying circuits. Order your copy today, only $4.95 postpaid.

● **The Semiconductor Memory Data Book for Design Engineers,** by Texas Instruments, Incorporated. Don't forget the importance of memories to your systems. Refer to this 272 page manual to find out about the T.I. versions of many of the popular random access memories and read only memories. Order your personal copy today, only $2.95 postpaid.

● **The Linear and Interface Circuits Data Book for Design Engineers,** by Texas Instruments Incorporated. When you run across one of those weird numbers like 75365 the immediate frustration problem occurs again. What kind of gate could that be? We won't tell in this ad, but you can find out by reading the specifications in **The Linear and Interface Circuits Data Book for Design Engineers.** You can interface your brain to the 72xxx (linear) and 75xxx (interface) series of functions by ordering your copy of this 688 page manual at only $3.95 postpaid.

● **The TTL Data Book for Design Engineers,** by Texas Instruments Incorporated. How does an engineer find out about the TTL circuits? He reads the manufacturer's literature. This 640 page beauty covers the detailed specs of most of the 7400 series TTL logic devices. No experimenter working with TTL has a complete library without **The TTL Data Book for Design Engineers.** Order yours today, only $3.95 postpaid.

Caution: Heavy Reading

Buyers of these books should be cautioned: heavy reading will be required. These books are so filled with information that they weigh in at a total of about 190 ounces (5387 grams). On the basis of sheer mass, these books have got to be the bargain of the century. Make sure that you use a structurally sound book shelf and above all avoid dropping one of these books on your foot. But the mass of these books doesn't affect the bargain: we pay postage on all orders shipped to addresses in the USA and Canada, so the prices you see are the prices you pay. (That's only $.005 per gram on the average.)

____ TTL Cookbook @ $8.95
____ TTL Data Book @ $3.95
____ Supplement to TTL Data Book @ $1.95
____ Linear and Interface Circuits @ $3.95
____ Semiconductor Memory Data @ $2.95
____ Transistor and Diode Data Book @ $4.95

Send to: Name ________
Address ________
City ____ State ____ Zip ____
□ Check enclosed
□ Bill MC # ____ Exp. Date ____
□ Bill BA # ____ Exp. Date ____
Signature ________

BYTE PETERBOROUGH, NH 03458

Join SCCS Now!!



The Southern California Computer Society is a non-profit, international association whose purpose is to promote and encourage a community interest in the computer arts and sciences. Of particular interest is the application of the computer to education and the social, scientific and environmental problems of society.

Formed in June 1975 by a handful of hobbyists and experimentalists, the society has grown rapidly to include 700 consumers, amateurs and professionals. The society serves as a focal point through which computer buffs meet, learn and share common interest and needs.

WHAT CAN SCCS DO FOR YOU?

- Frequent meetings for exchange of information and computer equipment.
- A magazine (SCCS INTERFACE) communicating topics and news of interest to all members in the community and their local organization activities.
- Classes on programming, building computers and general use of computers.
- Informal technical assistance and advice on anything related to computers.
- Consumer cooperative purchasing plans.
- A center under development with library, computer center and laboratoty.
- Receive 'Interface'— a monthly publication for the hobbyist and experimentalist.

WHO SHOULD JOIN SCCS?

- Anyone who is interested in computers as a hobby.
- Anyone who wants to be introduced to the world of computers (but doesn't know where to start).
- Anyone who has computer related talents to share.

DON'T MISS THE NEXT ISSUE OF INTERFACE

JOIN NOW • • • •

MEMBERSHIP FORM

SOUTHERN CALIFORNIA COMPUTER SOCIETY
P.O. BOX 987
SOUTH PASADENA, CA 91030
(213) 682–3108

NAME ______________________

ADDRESS ______________________

CITY ______________________

STATE ______________ ZIP __________

HOME PHONE (______) ______-__________

BUSINESS PHONE (______) ______-__________

DUES: $10.00/YEAR

PLEASE PRINT PLAINLY

MAKE CHECK PAYABLE TO: SOUTHERN CALIFORNIA COMPUTER SOCIETY



BOOK REVIEWS

Computer Models of Thought and Language *edited by Roger C. Schank and Kenneth Mark Colby. Published in 1973 by W.H. Freeman and Company, San Francisco CA.*

Some of the most difficult problems and the most exciting research areas in computer science are those which come under the heading of artificial intelligence. Contrary to the layman's science fiction imagery conjured by the name, artificial intelligence is a serious branch of research in the computer sciences. *Computer Models of Thought and Action* is a collection of papers by 10 different authors active in the AI field. It is a book which requires serious reading and is best approached by a person who has some experience in computer science and programming areas to act as a guide to understanding the material. To quote from the editor's preface:

> "The intent of this book is to describe attempts to model human psychological processes on a computer. Over the past ten years the field of Artificial Intelligence (AI) has become a recognized discipline in the academic and industrial communities. Much of the work in AI has been conceived to develop intelligent machines without claiming that the processes model human thought processes. Within AI, psychology and linguistics, however, there are growing groups of people concerned with representing human thought processes in the form of computer programs."

The content of the book reflects this editorial goal.

The introduction to the book is an excellent review article by Allen Newell, entitled "Artificial Intelligence and the Concept of Mind." In this article is found an excellent introduction to the field of artificial intelligence. The author provides a discussion of three different views of the artificial intelligence field: AI as the exploration of intellectual functions; AI as the science of weak methods; and AI as theoretical psychology. After treating these views of different approaches to AI, the article concludes on a philosophical note with a treatment of several questions on the nature of mind.

Then the book turns to the subject of natural language models. The idea of a computer which understands natural languages (like English) is an old science fiction theme. In this section of the book are found four detailed articles on theoretical and applied research into natural language analysis by computers:

> "Semantic Networks: Their Computation and Use for Understanding English Sentences," by R.F. Simmons.
>
> "An Artificial Intelligence Approach to Machine Translation," by Yorick Wilks.
>
> "A Procedural Model of Language Understanding," by Terry Winograd.
>
> "Identification of Conceptualizations Underlying Natural Language," by Roger C. Schank.

Here is a body of information on both theory and practice of natural language problems, information which is fundamental to work in these areas.

Following the section on analysis and utilization of information found in natural language situations, the emphasis of the book turns to models of belief systems. The purpose of a belief system as presented in "Simulations of Belief Systems" by Kenneth Mark Colby is to help explain the nature of human beliefs by constructing a computer

A Lesson in Economics

What happens when a good becomes so desirable and sought after that many people want to have it? Unless an ample supply is forthcoming, the price can and must go up (political fiat to the contrary notwithstanding). Recently we wondered why one distributor kept asking us for more and more of BYTE Number 1. He was very insistent, and spent several transcontinental telephone calls urging us (in vain) to send him more of BYTE's inaugural issue.

We found out recently, thanks to several callers, that this fellow was selling BYTE's Number 1 issue for $15.00 each, since everybody wants one and the supply is very limited. Of course he made a temporary windfall profit on his early investment in a supply of the new magazine. He paid what everyone else did who purchased a bulk shipment of the then unknown magazine. There is a law of economics that price rises when supplies are tight – it is as much a folly to fight such a trend as it is to attempt to legislate Pi = 3.0000... Many thanks to Adam Smith and Ludwig von Mises for tipping us off about the law of supply and demand, confirmed again by this laboratory experiment.

The point of mentioning this is to put to rest rumors about our supposed reprinting of the early issues of BYTE. First, we are not planning to reprint BYTE magazine issues as magazines, ever. You'll see individual articles reappearing in books of selected reprints which we plan to publish. However our early supporters will see their confidence rewarded as the price of the first issue collector's item rises over the years, since we will never reprint it as a magazine. Our policy is now and will continue to be one of matching our press runs to the subscription and newstand demand, with very few extra issues kept available. If you want to keep up with this fast moving field, you'll have to keep your subscription current.

system to simulate the process of judging credibility of statements. On a slightly different theme, Robert P. Abelson describes work with a computer simulation of the psychological characteristics of the "true believer." The first result of the simulation was "the Ideology Machine," a program developed to simulate attitudes characteristic of the true believer given a specific input model. The discussion of that model is a brief introduction to an extensive treatment of a redesigned model which grew out of the first attempt.

The final section of the book is devoted to models of memory and cognition. This includes "The Memory We Must Have" by Earl Hunt, "In Defense of Ad Hoc Systems" by Robert K. Lindsay, and "A Model for the Encoding of Experiential Information" by Joseph D. Becker.

The art of reading a book is one which requires careful thought and analysis by the reader if full utilization of the information is to be obtained. *Computer Models of Thought and Language* is a book which is designed for an advanced undergraduate or graduate level course in computer science, and thus requires full use of reading skills. It is not an easy book to read, and integration of its information will require careful thought on the part of the reader. But given these cautions, the book is an excellent source for some of the most sophisticated and interesting applications of computer technology. cth

When Harlie Was One *by David Gerrold, Ballantine Books, 1972, 279 pp. paperbound, $1.50.*

David Gerrold will be familiar to Star Trek (the series, not the game) fans as the author of *The World of Star Trek* and the "Trouble With Tribbles" episode. David Gerrold is also the author of one of the best stories on artificial intelligence to date.

David Auberson, the artificial intelligence psychologist in the novel, is instrumental in bringing the computer to life. (Care to define *life*, anyone?) The machine is a little different from conventional computers in that it has more than two states (0 or 1) to represent a decision. There is yes and no, of course, but with many levels of maybe in between. The pun on *H*uman *A*nalog *R*obot, *L*ife *I*nput *E*quivalents (HARLIE), i.e., David's son, is purely intentional.

As Harlie grows up (up?) he learns to transfer information via a communications network and causes some very interesting mistakes to be made in other computers. Some of the most interesting conversations in science fiction occur between the psychologist and Harlie, especially on such issues as love, death, the impact of computers on society and the impact of society on a poor thinking machine.

When Harlie Was One is one of the most delightful novels about computers around. Anyone even slightly connected to computers should find this book very entertaining.

The Shockwave Rider *by John Brunner, Harper & Row, 1975, 246 pp., $8.95.*

John Brunner is an outstanding science fiction author (Stand On Zanzibar) who takes the study of Alvin Toffler's *Future Shock* very much to heart, creating a future that is very fast and media bound. His recent novel, *The Shockwave Rider,* is a part of this vision that computer oriented readers should find very interesting. The book reveals a near future that has a complex computer network covering the North American continent with interconnected government and industrial data centers monitoring all of the financial, transportation, and status transactions of individual citizens. Access to this information is available if you know how to get it, and most citizens have become paranoid from knowing the information is there, but not knowing who will see it.

Our hero, a very bright specialized student with a high aptitude for programming, "drops out" and avoids the tracking and monitoring of the network by learning to change his identity completely by assigning himself new status, salary, and professions through some complicated maneuvers with network terminals. His travels lead him to a post disaster scene of the west coast area where the predicted earthquake has finally toppled the San Andreas fault. Isolated areas of the coast are still not under the influence of the network, and some of his talent is used by playing "fencing," a computer game of skill.

Our hero has the last laugh by writing "tapeworm" programs that grow through the network and are detectable, but very difficult to stop. One such tapeworm causes very embarrassing government data to be displayed in front of all citizens. Hmm . . . Computers for social reform, anybody?

The Shockwave Rider is very well written and should provide a very interesting scenario to those interested in issues of data access, invasion of privacy, etc. In other words, just about anybody.

Gary Liming
3152 Santiago Drive
Florrisant MO 63033

Continued from page 8

own back yard also argue for an organization which is national in scope. Is it time for such an organization in this area of small scale computing? Most likely yes: There is already indication that some form of nationwide affiliation is beginning to form.

I refer to that contradiction in terms, the *Delaware Valley* (i.e., Philadelphia area) chapter of the *Southern California* Computer Society, which is mentioned in the Clubs and Newsletters section of this issue of BYTE. (It would seem strange and somehow provincial for an organization aspiring to nationwide scope to retain the name Southern California Computer Society.) The idea is right, even if an appropriate name is not yet available. From the start, the people who have been running the Southern California Computer Society have been doing an excellent job with the programs which make up an active organization of this sort. This evaluation is based upon the SCCS *Interface* newsletters and upon phone conversations with individuals who are involved with SCCS.

Advantages of a Nationwide Affiliation

A computer society with nationwide membership and scope of operations can contribute the same sort of benefits which a local organization can, but the scope of the operation can produce some extra effects as well.

For instance, the idea of informational activities and seminars based upon taped lectures works better if coordinated through a nationwide organization. In my phone conversation with an SCCS person recently this was mentioned as one of the programs that was envisioned for the national organization. A nationwide organization can catalog and distribute tapes of lectures and talks, thus making available talent in all affiliated computer groups.

A national personal computing society could also be used to centrally coordinate numerous things ranging from the planning of manufacturers' tours and road shows to the continued nationwide group purchasing items like LSI-11 computers and hard copy terminals. The national personal computing society could also help coordinate standards for media interchange and data formats which are independent of the computer system being used. Working through such an organization, it might even be possible to arrive at some standards for computer languages appropriate for small scale computing's system programming needs.

Then of course there is the usual interpersonal friendship value of such an organization. As a friendship signal, such an organization would certainly require some form of logo which could be used universally. In decal form it could be applied to automobile windows; in lapel pin form it could be used to "break the ice" in meeting individuals outside of society meetings.

This is by no means a complete list of all the possibilities inherent in this idea of a national personal computing society. The full development of the idea requires more input and thought than I could possibly put into this short essay on the subject. My net conclusion is that readers interested in forming a computer group, and existing computer groups, should contact the SCCS at:

Southern California Computer Society
PO Box 987
South Pasadena CA 91030

The SCCS should be able to help get such a club started, and possibly will already have a local contact in your area if you are not the first to inquire.■

October BOMB Results

The winner of the $50 prize for the most popular article in the October 1975 BYTE is Don Lancaster for his article "Ins and Outs of Volatile Memory." Very close behind in the voting were Dan Fylstra's "Son of Motorola" and Peter Helmers' "Versatile ROM."

BOMB: BYTE's Ongoing Monitor Box

BYTE would like to know how readers evaluate the efforts of the authors whose blood, sweat, twisted typewriter keys, smoking ICs and esoteric software abstractions are reflected in these pages. BYTE will pay a $50 bonus to the author who receives the most points in this survey each month. (Editor Helmers is not eligible for the bonus.)

- *Articles you like most get 10 points, articles you like least get 0 (or negative) points – with intermediate values according to your personal scale of preferences, integers only.*
- *Only one entry per reader.*
- *Fill out your ratings, and return it as promptly as possible along with your reader service requests and survey answers.*

Page No.	Article	LIKED: LEAST … BEST
18	Grappel: My Dear Aunt Sally	0 1 2 3 4 5 6 7 8 9 10
26	Maurer: Processing Algebraic Expressions	0 1 2 3 4 5 6 7 8 9 10
32	Liming: Data Paths	0 1 2 3 4 5 6 7 8 9 10
42	Vice: The New Altair 680	0 1 2 3 4 5 6 7 8 9 10
46	McIntire: How to Save the BYTEs	0 1 2 3 4 5 6 7 8 9 10
50	Kay: More on SWTPC 6800 System	0 1 2 3 4 5 6 7 8 9 10
54	Gray: LEDs Light Up Your Logic	0 1 2 3 4 5 6 7 8 9 10
58	Walde: Build a TTL Pulse Catcher	0 1 2 3 4 5 6 7 8 9 10
62	Lancaster: TV Color Graphics	0 1 2 3 4 5 6 7 8 9 10
74	Warren: View From Silicon Valley	0 1 2 3 4 5 6 7 8 9 10
76	Rush: Could a Computer Take Over?	0 1 2 3 4 5 6 7 8 9 10
86	Helmers: Joysticks and Slide Pots	0 1 2 3 4 5 6 7 8 9 10

BYTE reader service

To get further information on the products advertised in this issue of BYTE merely tear, rip, or snip out this advertiser index, fill out the data at the bottom of the page, circle the appropriate numbers, and send the works to BYTE, 70 Main St, Peterborough NH 03458. Readers get extra Brownie Points for sending for information since this encourages advertisers to keep using BYTE — which in turn brings you a bigger BYTE.

ADVERTISER INDEX

Reader's Service
BYTE
Green Publishing Inc
70 Main St
Peterborough NH 03458

Please print or type.

FEBRUARY 1976

BYTE acquired via
☐ Subscription
☐ Newsstand
☐ Stolen

Name ____________________

Address ____________________

City __________ State ______ Zip ______

Coupon expires in 60 days . . .

THE BYTE QUESTIONNAIRE

BYTE is dedicated to the needs of its readership. In order to better gauge matters of editorial policy and content, as well as to give our advertisers some "hard facts," we publish this questionnaire.

Indicate your familiarity with computer languages.

Computer Language	Use it.	Know of it.	Never heard of it.
Assembly Languages	( )	( )	( )
BASIC	( )	( )	( )
COBOL	( )	( )	( )
FORTRAN	( )	( )	( )
PL/1	( )	( )	( )
BLISS	( )	( )	( )
PASCAL	( )	( )	( )
ALGOL	( )	( )	( )
XPL/PLM	( )	( )	( )
Other: ________	( )	( )	( )
________	( )	( )	( )

What is your equipment?

Television Interface ____________________

Processor ____________________

Memories ____________________

Audio Interface ____________________

Bulk Memory ____________________

Other ____________________

What software do you have for your machine?

Do you read science fiction? Who is your favorite author?

How did you find out about the world of small computers?

Feel free to photocopy this page if you wish to keep your BYTE intact.

CALL FOR PAPERS-ACM '76
OCTOBER 20-22, 1976...

HOUSTON

Our 1976 Annual Conference will be held October 20 – 22 in Houston. We feel that it will be one of the most relevant ACM conferences ever held, both for the practicing professional and the researcher. The conference facilities in downtown Houston's Regency Hyatt House will be outstanding. You will find that ACM '76 will be a place where you can enjoy yourself, renew old acquaintances, make some new ones, and participate in a serious technical discussion of the problems and potential of our industry. Plan to attend.

Olin Johnson
General Chairman, ACM '76

about the conference

You are invited to submit a paper on any aspect of computer research, development, or implementation. High quality papers of a theoretical, state-of-the-art or tutorial nature are welcome. The technical program for ACM '76 will again be organized around the ACM Special Interest Groups, although additional sessions will be provided for papers of general interest or those not related to any SIG.

Papers must not have been previously presented or published; they should not exceed 10 published pages, including a 100-word abstract, bibliography, and illustrations; and they must be received in four copies by March 1, 1976. All papers will be refereed either by the SIG's or by reviewers selected by the Technical Program Committee. Notification of acceptance will be made by May 1. If you intend to submit a paper, and we sincerely hope that you do, please send in the coupon below. If you have an idea for a good session or some other technical program idea, please do the same or start promoting your idea in your SIG –– but do it now! Be a part of the spirit of ACM '76!

about the technical program

ACM '76

Be a part of the "Spirit of ACM '76"

ASSOCIATION for COMPUTING MACHINERY
1976 ANNUAL CONFERENCE

Roger W. Elliott
Technical Program Chairman, ACM '76
Department of Industrial Engineering
Texas A & M University
College Station, Texas 77843 (713) 845-5531

Name ____________ Phone ____________

Address ____________

City ____________ State ____________ Zip ____________

☐ I intend to submit a technical paper.

Tentative title ____________

____________ SIG (optional) ____________

☐ I am enclosing a technical program suggestion.

☐ Please call me. I would like to discuss an idea for the technical program.

If Napoleon had owned an Altair.



Things might have turned out differently.